Effective techniques for handling incomplete data using decision trees

Thesis

EFFECTIVE TECHNIQUES FOR HANDLING INCOMPLETE DATA USING DECISION TREES

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF STATISTICS
AT THE OPEN UNIVERSITY, MILTON KEYNES
IN FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

By
Bhekisipho ETH Twala

January 2005

This thesis is dedicated to the memories of my dad and three sisters - Bhekithemba (1929-2002) Zanele (1966-1997), Hlengiwe (1971-1997) and Nokuthula (1973-2001)

Abstract

Decision Trees (DTs) have been recognized as one of the most successful formalisms for knowledge representation and reasoning and are currently applied to a variety of data mining or knowledge discovery applications, particularly for classification problems. There are several efficient methods to learn a DT from data. However, these methods are often limited to the assumption that data are complete.

In this thesis, some contributions to the field of machine learning and statistics that solve the problem of extracting DTs for learning and classification tasks from incomplete databases are presented. The methodology underlying the thesis blends together well-established statistical theories with the most advanced techniques for machine learning and automated reasoning with uncertainty.

The first contribution is the extensive simulations which study the impact of missing data on predictive accuracy of existing DTs which can cope with missing values, when missing values are in both the training and test sets or when they are in either of the two sets. All simulations are performed under missing completely at random, missing at random and informatively missing mechanisms and for different missing data patterns and proportions.

The proposal of a simple, novel, yet effective proposed procedure for training and testing using decision trees in the presence of missing data is the next contribution. Original and simple splitting criteria for attribute selection in tree building are put forward. The proposed technique is evaluated and validated in empirical tests over many real world application domains. In this work, the proposed algorithm maintains (sometimes exceeds) the outstanding accuracy of multiple imputation, especially on datasets containing mixed attributes and purely nominal attributes. Also, the proposed algorithm greatly improves in accuracy for IM data. Another major advantage of this method over multiple imputation is the important saving in computational resources due to it simplicity.

The next contribution is the proposal of three versions of simple probabilistic techniques that could be used for classifying incomplete vectors using decision trees based on complete data. The proposed procedure is superficially similar to that of fractional cases but more effective. The experimental results demonstrate that these approaches can achieve comparative quality to sophisticated algorithms like multiple imputation and therefore are applicable to all kinds of datasets.

Finally, novel uses of two proposed ensemble procedures for handling incomplete training and test data are proposed and discussed. The algorithms combine the two best approaches either with resampling (REMIMIA) or without resampling (EMIMIA) of the training data before growing the decision trees. Experiments are used to evaluate and validate the success of the proposed ensemble methods with respect to individual missing data techniques in the form of empirical tests. EMIMIA attains the highest overall level of prediction accuracy.

Acknowledgements

Thanks Lord for making all things possible and for all the blessings you have bestowed on me and my family. With every word on this thesis I give thee praise.

I always considered it trite and over-fulsome of thesis authors to list a full set of people who were 'invaluable'; always, that is, until I wrote a thesis and realised just how deeply I am in debt to the people listed below.

First and foremost, credits and very special thanks to my supervisor, Chris Jones, for willing to take a chance with me and making this venture a reality, and without whom this thesis would not have been possible. Writing this thesis was a blast. Our concept from the beginning was to prepare a thesis that was different and fun writing it. I feel like we succeeded on both counts. You are a master, as in academia... so in life!!! To thank you in perpetuity for all that you have done for me, I proudly name the principal construction of this thesis – the CMJ series – after you.

To David Hand, sincerest thanks for your great talents, instinct, constant patience, ideas, suggestions and great mind – it is indeed a pleasure to know you. Additional special thanks and great appreciation to Allan White for your tremendous help, thoughtful commentary, practical advice and for forcing me to think on my feet – it has been both enlightening and entertaining!!! I am also thankful to Wray Buntine for giving me a solid start, especially for encouraging me to learn about implementation properly; and as I do theory, whip up a quick implementation to back up things... hence, in part, bringing that magic to research.

I would like to express my thanks to the Open University for giving me an opportunity to do such a research I can be proud of and for contributing to a research environment that has proven immensely rewarding. To all the members and staff of the Statistics Department at the Open University - it never ceases to amaze me how much the unique and incredible talents of each individual contribute to a thesis. That helped bring this endeavour to life, thank you. I also thank all the members of the 'Thunder Birds Team' (TBT). There will probably never be a time where the

technology involved when writing a thesis will be snag free. I am so grateful that it is you guys that deal with it instead of me!!!

Much love and respect to my officemates and "sisters", Mona Kanaan and Jane Warwick (at least you can now think in S-PLUS!!!) for putting up with me for almost three years, and for enriching my life immeasurably simply by being such interesting and diverse people.

It gives me great pleasure to thank Martin Shepperd and Michelle Cartwright for your endless patience and goodwill and in letting me finish this project.

Several teachers of statistics have influenced me profoundly at pivotal moments in my early education. I wish to thank Antoni Szubarga for teaching me that the essence of statistics is ideas – not notation!!! I still refer to the copy of M.H. Degroot's book on "Probability and Statistics" which you gave me. I also wish to express warm thanks to M.A. Ali for showing me statistics the way it was meant to be done, and whetted my appetite for more. Like my current supervisor, you also gave me extraordinary freedom to pursue my own interests as a graduate student. It is no exaggeration to say that almost all of the results of this thesis are in some respect fruits of the seeds planted during my time at the University of Swaziland, and for this, I owe you a debt of gratitude. Additional thanks to Elkana (Ali) Ngwenya for teaching me by his own shining example that the most important step by far in any statistical problem-solving is finding the right point of view – the point of view that makes things simple!! In retrospect, you introduced me to the "work smarter, not harder" philosophy of statistics on which I have come to depend.

While at Salesian High School, I was fortunate to learn statistics from Eamon Molloy. I thank you for encouraging me to pursue independent research even at this early age, and for fostering my love of statistics for its own sake. Thank you, Eamon, for launching me on the journey of a lifetime. It was a privilege to be in your class and to have access to a statistician with a masters. I would also like to thank the late Fr. Luke Boyle for his generous nature and for nurturing my development as a high school student, and for enriching my life immeasurably.

Now that I have recounted the entire history of my statistical upbringing, I have the pleasure of acknowledging those who have provided me with the personal foundation which has been so indispensable to my professional life.

First and foremost, grateful thanks to my mom (Siphiwe) and late dad (Bhekithemba) whose love, understanding and bedrock of support has made my academic career possible - without you none of this would have been possible. You have always been the wind beneath my wings. Your patience and understanding with the relentless demands of graduate study will surely earn you a place among the Saints of the Ages!!!

Major domos to "my female team" at home; my wife and constant companion Ntonto, and two lovely daughters, Okuhle (OO) and Nobhekisipho (NoB) for putting up with me, and for dealing on a daily basis with a crazy husband and dad. However, there certainly is never a dull moment in our house. For the millionth time, thanks for your patient kindness and love, and for never raising an eyebrow when I claimed my thesis would be finished in the "'next two weeks" for nearly two years. You are all my reasons for living. I will always cherish you.

Much love and happiness back to you, each of my sisters and all our relations; and to all my nephews and nieces, wherever you are, "I love you!!"

Finally, I also remain truly grateful for the wealth of support and talent I shared with so many researchers, friends, and colleagues. Thanks for bringing gourmet dishes to the research table. You are as solid as they come, not only in your craft, but as people.

Is that everyone??

I alone remain responsible for the content of the following, including any errors or omissions which may unwittingly remain.

"Sometimes a scream is better than a thesis."

~Ralph Waldo Emerson

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction and Background

Machine Learning (ML), which has been making great progress in many directions, is the hallmark of machine intelligence just as human learning is the hallmark of human intelligence. The ability to learn from observations and experience seems to be crucial for any intelligent being. Likewise, ML plays a central role in Artificial Intelligence research (Holland, 1975; Winston, 1992; Patterson 1996) and has reached a certain level of maturity. It has been heralded as the next revolution in learning systems by some experts in the area--but dubbed an oxymoron by others. ML is inspired by work in several disciplines. These include computer science, mathematics, psychology, (neuro) biology/genetics, philosophy and among other areas.

Common goals and similar evaluation methods drive ML research. The main aim is to improve performance on some task, and the general approach is finding and exploiting some regularities in training or learning data. The goals driving ML research can be psychological (understanding human learning); empirical (to discover principles relating algorithm characteristics); mathematical (to analyse what concepts are learnable at all); and applications (to use the learning system and results of learning in a real-world situation).

Recently, ML research (Hart, 1984; Michalski, 1986; Forsyth and Rada, 1986; Winston, 1992; Michie *et al.*, 1994; Langley, 1996; Mitchell, 1997; Barry and Linoff, 1997) has begun to pay off in many ways. ML methods are being successfully integrated with powerful performance systems and more established techniques have already made their presence felt. Recent successes in ML and statistics include neural network learning, Bayesian network learning, instance-based or case-based learning, genetic algorithm learning, analytic learning and rule induction, and so on. To date the above could be identified as the six basic ML paradigms under active investigation. These paradigms emerged from different scientific roots; differ in their

assumptions about the representation, performance and assessment methods, and algorithms used in a learning system. They employ different computational methods, and often rely on subtly different ways of evaluating success, although all share the common goal of building machines that can learn in significant ways for a wide variety of task domains.

ML paradigm techniques are also being applied to new kinds of problems including knowledge discovery in databases, language processing, robot control, and combinatorial optimisation as well as more traditional problems such as speech recognition, face recognition, handwriting recognition, medical data analysis, game playing, discrimination, classification, and so on. Some ML paradigms, like rule induction, which employ decision trees (DTs), are specifically designed to deal with uncertainty and are currently applied in a variety of data mining/knowledge discovery/statistical modelling applications, particularly for classification problems. These are the type of problems that are covered in this thesis.

Classification rules (or classifiers) are used to predict the corresponding class of an example, where the class is some discrete variable of practical importance. An application of classification using decision trees is when a bank makes a credit decision or considers giving loans to its customers. This is done by a set of rules that classify each loan applicant as low, medium or high risk, using data collected about previous customers together with if the loans were good, bad or worse. Such rules are called classification rules and the data on old customers is called the training sample of classified cases (training set) from which the classification rules are discovered. These classification rules can then be used to discover the group a new customer belongs to. The main question is whether the objects fall into groups or clusters, as against more haphazardly scattered over the domain of variation.

Despite the success of ML research, there are some problems that currently exist within the ML community. These include bias, overfitting, noise, incomplete data, uncertainty, learning as search, and so on. Some of these problems are discussed briefly in Section 1.3.

1.1 The Meaning of Learning

To learn means to add something to a body of knowledge, a body that in the extreme might be empty. Learning can be thought of as an interaction between two logically distinct entities, the learner, **L**, and the supervisor or teacher, **T**. Learning situations differ in the degree to which the responsibilities of these two entries are functionally separate. The responsibilities of **T** can be thought of as providing examples or occasions for **L**, providing a feedback concerning the correctness of **L**'s response to an example and providing a sequence or ordering in which the examples are to be presented to **L**. The responsibilities of **L** can be thought of as providing a response or answer to each example presented or modifying its knowledge appropriately based on feedback from **T** concerning the correctness of its answer. Without learning, everything is new. Hence, it could be argued that a system that cannot learn is inefficient because it re-derives each solution and repeatedly makes the same mistakes.

A learning system uses sample data to generate an updated basis for improved performance on subsequent data from the same source, and expresses the new basis in intelligible symbolic form (Michie *et al.*, 1994). Early attempts to devise learning systems during the cybernetic days of AI proved disappointing, so the whole idea was dropped. Only recently has it been revived. Most, however, would still agree that ML has not lived up to the hype and expectations that began a few years ago even though there have been some advances in this field.

When a computer system improves its performance, without re-programming, it can be said to have learned something. Two of the most central goals of any learning algorithm are to provide more accurate solutions and to cover a wide range of problems that seem to require intelligence, respectively. Other minor goals could be to obtain answers economically and to simplify codified knowledge (Forsyth *et al.*, 1994).

All systems designed to modify and improve their performance share important features. A typical machine learning system does not interact directly with its

environment. It uses "coded" observations of this environment to learn about it. Fig. 1.1 depicts a framework for a typical machine learning system. The environment *E* represents the real world, the environment that is learned about. *E* represents a finite number of observations, or objects, that are encoded in some machine learning readable format by the encoder *C*. The set of encoded observations is the training set for the learning algorithm *ML*.

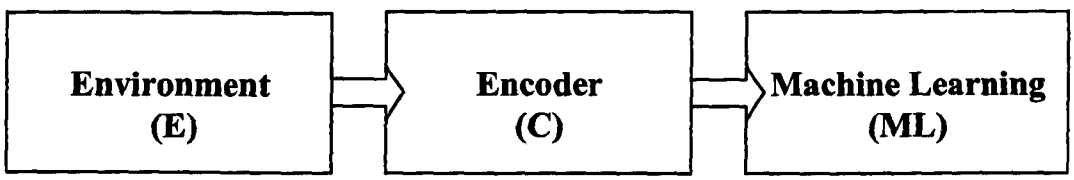

Fig. 1.1. machine learning diagram

Most ML algorithms use a training set of examples as the basis for learning. During the learning phase, the system does not interact directly with its environment (also called a model or a classifier for classification problems), but uses coded observations, often stored in a set of labelled examples - called the training set. The notion of a training set is important in understanding how a machine learning system is tested. Typically there is a database of examples for which the solutions are known. The system works through these instances and derives a rule or set of rules for associating input descriptions with output decisions (Forsyth and Rada, 1986).

The general framework for a tree classifier, Figure 1.2 is a variation of the machine learning framework.

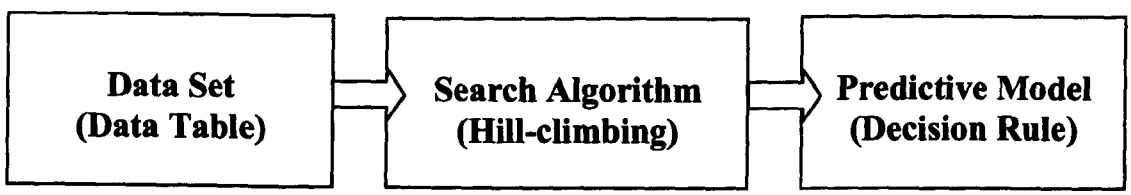

Fig. 1.2. A decision tree framework

The first step is to select the types of data that will be used by the mining algorithm (decision tree). The search procedure is part of the system that carries out the task.

It explores the search space defined by a set of possible representations. Associated with each search strategy is the evaluation component or function that can be used to estimate the distance from the current state to the goal state. If these estimates are computed at each choice point, then they can be used as a basis for choice. It is worth mentioning that such a type of framework could apply within different research communities. For an example, statisticians would allow models to be used for estimation purposes (as a search procedure), like, utilising the maximum likelihood (ML) for estimating missing values. Recently, Hand *et al.* (2001) have shown how such a framework could apply for data mining.

Two learning techniques are of special interest. In supervised learning, the system searches for descriptions for the user-defined classes (training examples with the correct classification for each example are given). In unsupervised learning the system constructs the summary of the training set as a set of newly discovered classes, together with their descriptions (training examples without classifications are given).

1.2 The Philosophy of Induction

Before discussing further how systems are made to learn it is important that the philosophers' and psychologists' thoughts about the subject be looked at.

Most learning systems are based on the general principle of induction, i.e. the reasoning technique to infer information that is generalised from the information in the database.

Both philosophers and psychologists have formulated laws by examination of several pathological situations caused by multiple comparison procedures (induction). They have looked at the part played by inductive reasoning in knowledge discovery and have attempted to find rational grounds to validate them.

A typical act of induction is one about treatments used in the medical community, whereby scientific evidence comes from clinical trials, but then an induction step is needed to argue that the drug can help people who are not in the trial.

Another act of induction is the psychic watch repair. A self-proclaimed psychic, Uri Geller, claimed to start previously-broken watches running using his psychic powers. A wonderful story has been heard (though not been able to verify) about how he asked the listening audience of a radio show to place watches on the radio and he would reach out with his psychic powers and start them running. Many listeners then called the station to say that their watches had started to run. However, the number who tried and failed is not known nor the number who could have placed their watch on the radio at another time and had it start running.

Another old favourite is the sunrise "problem"; the sun always rises from the East and set in the West. Never in living memory has anyone seen it do anything else. Throughout recorded history it has always risen in the East and set in the West. Thus, it will do so tomorrow as well.

Hidden prophecies in text is another act of induction. Recent controversy over the book "The Bible Code" has shown that searching over many possible "skip codes" can find apparent hidden messages in any text, not just the bible. Examples are the Microsoft License Agreement and Tolstoy's War and Peace.

Lateral thinking could also be an act of induction, especially to human-problem solving. A good example is of a merchant who owes money to a money lender and agrees to settle the debt based upon the choice of two stones (one black, one white) from a money bag. If his daughter chooses the white stone, the debt is cancelled; if she picks the black stone, the moneylender gets the merchant's daughter. However, the moneylender "fixes" the outcome by putting two black stones in the bag. The daughter sees this and when she picks a stone out of the bag, immediately drops it onto the path full of other stones. She then points out that the stone she picked must have been the opposite colour of the one remaining in the bag. Unwilling to be unveiled as dishonest, the moneylender must agree and cancel the debt. The daughter has solved an intractable problem through the use of lateral thinking.

Even though both acts of induction are of common sense, that does not mean they are necessarily valid - not because the sun would be expected to rise from the West and set in the East or not expect watches to run because of someone's psychic

powers. Crucial assumptions are needed to justify them. This is one important reason why many philosophers are devoting a lot of attention to the problems of induction these days.

"...it is the peculiar and perpetual error of human intellect to be more excited by affirmatives than by negatives; whereas it ought properly to hold itself indifferently disposed towards both alike. Indeed in the establishment of any true axiom, the negative instance is the more forcible of the two" (See Hart, 1984)

The problem addressed by an inductive-learning system (as shown in Figure 1.2) is to take a collection of labelled "training" data and form rules that make accurate predictions on future data. Inductive learning is particularly suitable in the context of an automated design system because training data can be generated in an automated fashion. For example, one can choose a set of training goals (a training goal is a design goal used for training purposes) and perform an optimisation for all combinations of training goals and library prototypes.

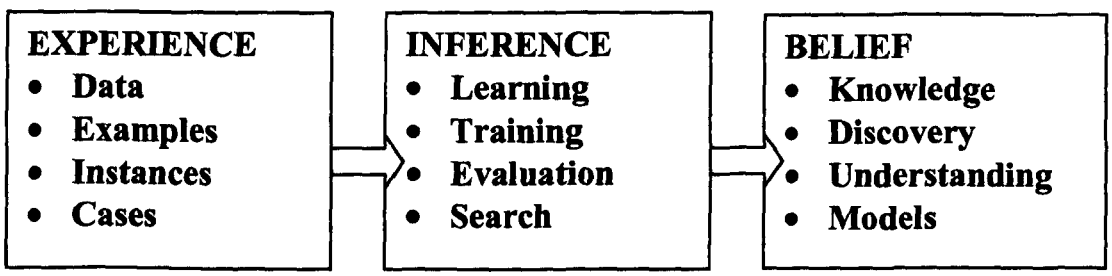

Fig. 1.3. An induction learning framework

One can then construct a table that records which prototype was best for each training goal. This table can be used by the inductive-learning algorithm to generate rules mapping the space of all possible goals into the set of prototypes in the library. If learning is successful, this mapping interpolates or extrapolates from the training data and can be used successfully in future design sessions to map each new goal into an appropriate initial prototype in the design library.

1.3 Problems in Machine Learning

There have been a variety of problems that have been the focus of research in the ML community. The most commonly referred to problems shall now be reviewed, even though much work has been spent during the last years to handle them. Also, some of these problems are strongly related to each other, and these relationships have led to considerable attention being devoted to them.

1.3.1 Noise and Overfitting

One problem is that large amounts of data are needed for inductive learning. In many real problems there is a degree of uncertainty or error (imperfection) present in the data. These could lead to errors in the classification process. One source of uncertainty is that of random errors or "noise" which is inevitable. There are many kinds of "noise" that could occur in the examples. These include errors, spurious correlations (i.e. correlations that are due mostly to the influences of one or more "other" variables), attributes that are not recorded, two examples having the same attribute/value pairs but different classifications, some values of attributes being incorrect because of errors in the data acquisition process or the processing phase, values of attributes being missing, and the classification or class label being wrong (for example, 1 instead of 2) because of some error. Monago and Kodratoff (1987) present a more detailed analysis of the sources of noise in data.

The next problem is also related to unnecessary attributes (which can be caused by noise) which, besides making no contribution to the predictive performance of the learning system, will simultaneously impose an extra computational burden. This situation is generally referred to as overfitting (Schaffer, 1993; Forsyth *et al.*, 1994; Cohen and Jensen, 1997), i.e. overfitting the training example data. For example, if the hypothesis space has many dimensions because of a large number of attributes, meaningless regularity maybe be found in the data that is irrelevant to the true, important, distinguishing features. Overfitting is harmful for several reasons. First, overfitted models are incorrect; they indicate that some variables are related when they are not. Second, overfitted models are difficult to understand due to the

unnecessary component that complicates attempts to integrate induced models with existing knowledge derived from other sources, and overfitting avoidance has sometimes been justified solely on the grounds of producing comprehensible models. Finally, overfitted models can have lower accuracy on new data than models that are not overfitted as demonstrated with a variety of domains and systems by Quinlan (1987).

In the area of decision tree learning, overfitting is avoided or fixed, to a certain extent, by: 1) Termination of tree growth when further splitting the data does not yield a statistically significant improvement or by 2) Growing a full tree, then pruning the tree by eliminating nodes. In practice the latter approach has been more successful.

1.3.2 Missing Values

Another problem that is related to noise is that of incomplete data or missing values. The presence of missing values is commonplace in large real-world databases. This has become one of the most important problems in academic research since most learning systems and statistical analysis were in the early stages not designed to handle missing data (incomplete vectors). There are several reasons why there are missing values in data. An item could be missing because it was unavailable or arises by "default" in data recording activities. Missing values could also occur because of confusing questions in the data gathering or because of sensor malfunction. In some situations the missingness could be caused by the relationships between the attribute variables themselves. That is, the information that is missing on a given attribute variable could be as a result of its relation to values of other attribute variables in the data set. An extreme case is that the missing value could be as result of its relation to an unobserved (missing value) in the data set.

Both large and small amounts of missing and/or faulty details in data might mislead the learning process and have an influence in various statistical measures, yet not a lot of work has been done in the research field.

1.3.3 Bias

The importance of bias has received a lot of attention over the last few years.

By definition, the bias of an estimator is the difference between its mean and the true value. Suppose that there is a function $f(x) = Y$ to learn given some sample (X, Y) pairs. There are several hypotheses that could be made about this function. A preference for one hypothesis over the others reveals the bias of the learning system. For example to prefer piece-wise functions or prefer smooth functions or prefer simple functions and treat outliers as noise.

Utgoff (1996) introduced several notions of biases. These are: good bias (which is appropriate to learn the actual concept), strong bias (restricts the search space considerably but is independent of appropriateness), declarative bias (defined declaratively as opposed to procedurally) and preference bias (bias that is implemented as soft preference rather definite restrictions to the search space). Extensions to the problem of bias can be found in Haussler (1988) and Ripley (1996).

In medical statistics bias could define a systematic disposition of certain trial designs to produce results consistently better or worse than other trial designs. Hence, wherever bias is found it results in a large over-estimation of the effect of treatments. For example, a poor trial design makes treatments look better than they really are. It can even make them look as if they work when actually they do not work. This is why good guides to systematic review suggest strategies for bias minimisation by avoiding including trials with known sources of bias.

1.3.4 Learning and Search

This is one other important problem that has to be taken into account when developing learning search systems. Search is a fundamental problem solving technique employed by human beings and also by computers. As a search problem, a search through all possible functions or rules is carried out to see which best accounts for the example data. Whenever there are several possibilities to continue

from a certain point and there is no way to determine in advance what possibility will lead to the intended goal then a search is performed.

Search problems are common place in ML research. Usually the search approaches employed by computer systems are not the same problem solving approaches that people use and therefore it is not obvious how the learning process of people can be transferred to the search approaches. It is also unclear within the ML community as to what set of goals a learning system should be searching for. In fact, in general the 'search space' is very large and impractical. So, inductive learning is about clever ways of managing search for possible rules. This search problem arises in DT induction during the splitting and/or pruning stages.

1.3.5 Other Problems

There are some other common problems in ML research like residual variation (where extraneous factors that are not recorded affect the results leading to unexplained variability in terms of the available data). There is also the problem of how to integrate already existing background knowledge into the learning process. Another problem is, if a system learns only from what the system has already seen, there is no guarantee that what has been learned will be correct for all future unseen situations.

1.4 Aims and Outline of the Thesis

This thesis addresses the problem of dealing with missing data in the context of supervised learning for classification tasks using decision trees. The attractiveness of decision trees is due to the fact that, in contrast to other tools for classification and prediction, decision trees represent rules. Rules can readily be expressed so that humans can understand them or they can even directly be used to discover regularities and improved future decisions by using historical data (data mining, knowledge discovery from databases, machine learning or advanced data analysis). Hence, decision trees could be considered as one of the generation of data mining algorithms.

Unfortunately, when constructing tree classifiers one must deal with the problem of missing data (incomplete attribute vectors). New techniques for rule extraction with DTs for classification tasks from possibly incomplete databases are explored and developed and are shown to perform reasonably well compared to other approaches previously proposed. For tree classifiers, two cases can be distinguished: first, growing (training) trees from incomplete data; secondly, classifying (testing) a new incomplete vector.

In the introduction the issues and importance of ML research and the most prominent paradigms were reviewed. The most commonly referred to problems in the ML community were also looked at. The conclusions in Chapter 8 shall give the reader some answers to the open questions that had been raised during the thesis work.

Chapter 2 contains a basic introduction to relevant methods of supervised learning and how such methods have been used for classification tasks. The main focus is on tree classifiers. Current leading tree learning methods such as Classification and Regression Trees (CART) and C4.5 and other methods are reviewed.

The missing value problem and the important definitions of missingness in data are presented in Chapter 3. The second part of Chapter 3 reviews tree learning techniques that have been used for dealing with missing values for both the training and test cases. The thesis to this stage consists almost entirely of review material.

A number of experiments and ideas are reported in Chapter 4. Twenty one datasets are used for these experiments. The first part of Chapter 4 starts with an experiment comparing the performance of various current techniques used for handling incomplete data given various missing proportions on both training and test data, and using three missing data mechanisms. The effect that incomplete training data have on predictive accuracy is explored in the second part of Chapter 4. The results of the simulation when looking at the impact of missing values when they occur only in the test set are presented in the third part of Chapter 4. The chapter is closed with a discussion of results.

In Chapter 5 an idea of constructing tree classifiers using incomplete training vectors and classifying incomplete vectors using trees is proposed. The performance of the new idea is compared with two current approaches previously proposed to deal with the problem and which achieved higher accuracy rates in experiments carried out in Chapter 4.

Chapter 6 presents a new probabilistic technique for classifying incomplete vectors. This new technique is in a form of three probability estimation methods. Once again, a series of experiments are carried out to analyse the performance of the new methods with the two best techniques previously proposed approaches to deal with the problem of incomplete test data (based on the simulation study carried out in Chapter 4).

Novel uses of two new ensemble procedures for handling incomplete training and test data are proposed and discussed in Chapter 7. Experiments are used to evaluate and validate the success of the new ensemble methods with respect to individual missing data techniques in the form of empirical tests.

Chapter 2

Classification and Decision Trees

Classification has two meanings in the statistical literature. It is concerned with assigning a sample to one of a set of previously recognised classes, on the one hand, and the construction and description of the classes themselves, on the other hand (Gordon, 1981; Gower, 1998). There are therefore two types of classification problems; supervised classification and unsupervised classification.

In supervised learning, for multivariate data, a classification function $y = f(x)$ from training examples of the form $\{(x_1, y_1),...,(x_m, y_m)\}$, predicts one (or more) output attribute(s) or dependent variable(s) given the values of the input attributes of the form $(x, f(x))$. The x_i values are vectors of the form $\{x_{i1},...x_{in}\}$ whose components can be numerically ordered, nominal, or ordinal. The y values are drawn from a discrete set of classes {1,..., K} in the case of classification. Depending on the usage, the prediction can be "definite" or probabilistic over possible values. Given a set of training examples and any given prior probabilities and misclassification costs, a learning algorithm outputs a classifier. The classifier is an hypothesis about the true classification function that is learned from, or fitted to, training data. The classifier is then tested on test data.

In contrast, the aim of unsupervised learning (clustering) is to formulate a class structure, i.e., to decide how many classes there are as well as assigning instances to their appropriate classes. For unsupervised classification, there is no such model and the number of classes (clusters, categories, groups, species, types, and so on) and the specifications of the classes are not defined or given. This type of classification problem identifies occurring structures in nature and it is variously known as cluster analysis (Wishart, 1999), Bayesian clustering (Stutz and Cheeseman, 1995), mixture modelling (McLachlan and Basford, 1988; McLachlan and Peel, 2000), intrinsic classification (Wallace, 1998) numerical taxonomy (Cohen and Martin,

1997), vector quantization (Gammerman *et al.*, 1995), unsupervised pattern recognition and unsupervised classification in different disciplines (Ripley, 1996).

This thesis focuses on supervised learning.

The most common types of predictor variables are now briefly defined.

A variable is continuous if its values are ordered, without a set of predefined values. Optionally a lower or upper bound can be specified for it. An example for such a variable is distance, which take values from a certain range. A discrete variable is characterized by predefined sets of numerical values. An example for a discrete variable is age. This variable takes values from the set {0, 1, 2, ..., 130}; A variable is nominal if it takes values in a finite set not having any natural ordering such as hair colour, marital status, and so on. A list of students in alphabetical order, a list of favourite cartoon characters, or the names on an organizational chart would all be classified as ordinal data. Boolean variables are binary and only have one possible cutting point. Because of this fact, it does not matter whether they are treated as ordered or nominal variables; A variable is ordinal if it takes values in a finite set whose values possess a clear ordering from greatest to lowest but the absolute distances among them is unknown. For example, Likert scales, Thurstone technique, preference scale, severity of an injury, and so on.

A wide range of algorithms in both classical statistics and from various ML paradigms have been developed for this task of supervised classification. These methods include linear or logistic discriminant analysis (Fisher, 1936; Cox, 1966; Hand, 1981; Krzanowski, 1990; McLachlan, 1992), logistic regression (Hosmer and Lameshow, 1989; Agresti, 1990; McCullagh and Nelder, 1990; Collett, 1991), density estimation (Silverman, 1986; Wand and Jones, 1995), memory-based reasoning which consists of variations on the nearest neighbour techniques (Cover and Hart, 1967; Dasrathy, 1991; Hand, 1997), DTs (Breiman *et al.*, 1984; Quinlan, 1993), neural networks (Ripley, 1996; Patterson, 1996), genetic learning (Grefenstette, 1991; Koza, 1992), association rules (Pearl, 1988; 1994; Agrawal *et al.*, 1993), rule induction (Clark and Boswell, 1991; Cohen, 1995), support vector machines (Vapkin, 1995; Burges, 1998), naïve Bayes classifier (Kononenko, 1991; Langley and Sage,

1994; Zheng and Webb, 1997) and so on. These methods are described in the following sections.

2.1 Discriminant Functions

Discrimination is the assignment of samples or the process of deriving classification rules from samples of classified objects. This assignment of samples can either be probabilistic or non-probabilistic (Gower, 1998). Distances between pairs of observations, between pairs of populations or between an observation and a population form the basis of many methods of multivariate analysis. The Mahalanobis and Euclidean distances are the two most widely used, especially for continuous data. Discrimination functions, which have received lots of attention in recent years, are one type of rules that are based on the Mahalanobis distance.

Discriminant Analysis (Fisher, 1936; Mardia *et al.*, 1979; Kendall, 1980; Hand, 1981; 1982; Dillon and Goldstein, 1984; Krzanowski, 1990; Everitt and Dunn, 1991; McLachlan, 1992) is a parametric method that is concerned with the problem of allocating an individual to one of the set of populations, on the basis of knowledge about those populations derived from the samples. Its main use is to predict membership in two or more mutually exclusive groups from a set of predictors, when there is no natural ordering on the groups.

Given a partition space (Ω), let $P(C_i|\tilde{x})$ denote the probability that a feature vector $\tilde{x}$ belongs to class i denoted C_i. Any function that computes $P(C_i|\tilde{x})$ is denoted a discriminant function (Duda and Hart, 1973; Hand, 1981; 1997).

Bayes theorem is applied to compute $P(C_i|\tilde{x})$.

$$P(C_i|\tilde{x}) = \frac{P(\tilde{x}|C_i)P(C_i)}{P(\tilde{x})} \qquad (2.1)$$

$P(C_i)$ are known as the *prior* (or *a priori*) probabilities and $P(C_i|\tilde{x})$ the *posterior* (or *a posteriori*) probabilities.

Since we have the information on the observations to be classified, namely a vector $\tilde{x}$, we can compare the probabilities of belonging to each class at $\tilde{x}$ and classify according to whichever is the largest

$$P(\tilde{x}|C_k)P(C_k) > P(\tilde{x}|C_j)P(C_j), \forall j \neq k \Rightarrow \tilde{x} \in \Omega_k \qquad (2.2)$$

where $\tilde{x} \in \Omega_k$ means that the object will be classified as belonging to class C_k. This rule is known as Bayes minimum rule (Hand, 1981).

$P(\tilde{x})$ can be ignored since it is the same for all the classes, and does not affect the relative values of their probabilities. A large number of assumptions are made in order to compute the conditional probabilities $P(\tilde{x}|C_k)$. These conditional probabilities are usually unknown but can be estimated from a set of classified samples.

Based on different assumptions made, discriminant functions can be defined in various degrees of polynomials such as linear and quadratic. DA has also been extended to quadratic discriminant analysis (QDA) and logistic discrimination. However, QDA is not covered in this thesis.

2.1.1 Linear Discriminant Analysis

The two most important assumptions in linear discriminant analysis (LDA) are that the data (for the variables) represent a sample from a multivariate normal distribution and the variance/covariance matrices of variables are homogeneous across groups (they are equal). With these assumptions, a linear discriminant function can be computed.

In order to understand how the *posterior* probabilities are computed for classification purposes, it is important to first consider the so-called Mahalanobis distance (a measure of distance between two points in the space defined by two or more correlated variables). Mahalanobis distance is used to do the classification, and thus, derive the probabilities.

Let $N_p(\mu, \Sigma)$ be the probability density function of the normal distribution, $\delta^2(x, y) = (x - y)'\Sigma^{-1}(x - y)$, for $x, y \in \Re^p$ is called the Mahalanobis distance between x and y in the p-dimensional space $\Re^p$.

Suppose that $P(C_i)$ is the prior probability of class C_i and that $f_i(x)$ is the normal probability density function of $\tilde{x}$ associated with class i, using the normal density function $N_p(\mu_i, \Sigma_i)$, $i = 1, 2$ with $\Sigma_1 = \Sigma_2$ and taking the Mahalanobis distance. The joint probability of observing class i and example x is $P(C_i) * f_i(x)$ and is given as follows:

$$P(C_i)\exp\left\{-\frac{1}{2}(x - \mu_i)'\Sigma^{-1}(x - \mu_i)\right\} \tag{2.3}$$

which reduces to

$$P(C_i)\exp\left\{-\frac{1}{2}\mu_i'\Sigma^{-1}\mu_i + x'\Sigma^{-1}\mu_i\right\} \tag{2.4}$$

times a term constant over i; where μ's are mean vectors and Σ's are covariance matrices.

Suppose there are 2 classes. Once the LDF has been calculated an observation x can be allocated to one of 2 populations on the basis of its "discriminant score". The discriminant function is taken to find the *a posteriori* class probabilities $P(C_i \mid x)$ given by equation 2.4.

The parameters μ_i and Σ are both unknown but they can be estimated using the sample mean vector of the ith class and the pooled estimate of the covariance matrix, respectively. The end result is that an observation is classified as belonging to class 1 if it has the highest posterior probability and to class zero otherwise.

When there are more than two classes, multiple discriminant analysis (MDA) is performed by estimating more than one discriminant function like the one presented above, requiring both independent and joint interpretation. For example, when there

are three classes, one model could typically be a good fit, say, for a function discriminating between class 1 and classes 2 and 3 combined. The other function will be a good fit for discriminating between class 2 and class 3. Therefore, taken in combination with one another, both of the functions explain the three classes better than either one would if interpreted alone. The order in which classes are partitioned is determined by the respective functions. Canonical analysis (McFadden, 1976; Krzanowski, 1990; McLachlan, 1992) has also been performed when dealing with discriminant functions for multiple classes.

LDA has the strength of having a form of classifier that is simply interpretable. Also, LDA is a one-step procedure that does not make a recursive partitioning of input space. One other strength of linear discriminant classifier lies in its ability to generate decision surfaces with arbitrary slopes. However, the assumptions made impose restrictions to problems to which LDA is applied. But it is known that, despite these restrictions, the linear discriminant function still performs well on data which do not satisfy the multivariate normality assumption and where the classes have different covariances (Ripley, 1996). A major drawback of LDA is its dependence on a relatively equal distribution of class membership. For example, if one class within the population is substantially larger than the other class, as it is often the case in real life, LDA might classify all instances in only one class. Also, LDA has the limitation of not being designed to handle categorical independent variables. Instead it codes the categorical variables into dummy variables. Its inability to treat missing values naturally has also been criticized (Breiman *et al.*, 1984). There is some evidence that the use of discriminant function estimation may tend to generate substantial bias in some applications (McLachlan, 1992).

2.1.2 Logistic Discriminant Analysis

Logistic discrimination analysis (LgDA), due to Cox (1966) and Day and Kerridge (1967), is related to logistic regression. The dependent variable can only take values of 0 and 1, say, given two classes. This technique is partially parametric, as the probability density functions for the classes are not modelled but rather the ratios between them.

Let $y \in \{0,1\}$ be the dependent or response variable and let $x = x_{i1}, x_{i2}, ..., x_{ip}$ be the predictor variables vector. A linear predictor η_i is given by $\beta_0 + \beta' x$ where β_0 is the constant and β' is the vector of regression coefficients $(\beta_1, ..., \beta_p)$ to be estimated from the data. They are directly interpretable as log-odds ratios or in terms of $\exp(\beta')$, as odds ratios.

The *a posteriori* class probabilities are computed by the logistic distribution:

$$P(y = 1 \mid x_{i1}, ..., x_{ip}) = \pi_i = \frac{\exp\{\eta_i\}}{1 + \exp\{\eta_i\}} \tag{2.7}$$

β' are estimated by maximising the likelihood function

$$L(\beta_0, ..., \beta_p) = \prod_{i=1}^{n} \pi_i^{y_i} (1 - \pi_i)^{1-y_i} \tag{2.8}$$

Computational details can be found in (Menard, 1995).

The estimated predicted value $\hat{\eta}_j$ and the estimated probability $\hat{\pi}_j$ for a new observation $x_{j1}, ..., x_{jp}$ are given by

$$\hat{\eta}_j = \hat{\beta}_0 + \hat{\beta}' x \text{ and}$$

$$\hat{\pi}_j = \pi(x, \hat{\beta}) = \frac{\exp\{\hat{\eta}_j\}}{1 + \exp\{\hat{\eta}_j\}}. \tag{2.9}$$

These terms are often referred to as "predictions" for given characteristic vector x. Therefore, a new element is classified as 0 if $\pi_0 \leq c$ and as 1 if $\pi_0 > c$, where c is the cut-off point score. Typically, the cut off point used could be 0.5 (Rumelhart *et al.*, 1986). In fact, the slope of the cumulative logistic probability function is steepest in the region where $\pi_i = 0.5$ (Pinder, 1996).

One advantage of using the LgDA (rather than LDA) is that it is relatively robust, i.e., many types of underlying assumptions lead to the same logistic formulation. By contrast the LDA approach is strictly applicable only when the underlying variables are jointly normal with equal covariance matrices. As with both LDA and QDA, one

difficulty of LgDA is its failure to deal well with categorical predictors, as these are transformed into dummy vectors. Thus, a disproportionately large number of degrees of freedom may be wasted. In terms of computational time LDA has been found to have a definite advantage over LgDA as it does not require the use of an iterative algorithm.

Logistic Regression (at least in the "logistic discriminant analysis" form) is another kind of supervised classification method. It is also a part of a category of statistical models called generalised linear models (Cox and Wermuth, 1966; McLachlan and Besford, 1988; McCullagh and Nelder, 1990; Menard, 1995).

2.1.3 The Multinomial Logit Model

The generalisation of the logistic discrimination approach to the case of three or more classes is known as the multinomial logit model (MLM) and the derivation is similar to that of the logistic discrimination model. To give a flavour of how this model can be used for classification, the procedure for a three-class case is sketched out. In this case, the probabilities of an observation belonging to each of the three classes, given a particular characteristic vector, are given by the following expressions:

$$p(\pi_1 \mid x) = \frac{\exp\{\hat{\eta}_1\}}{1 + \exp\{\hat{\eta}_1\} + \exp\{\hat{\eta}_2\}}$$

$$p(\pi_2 \mid x) = \frac{\exp\{\hat{\eta}_2\}}{1 + \exp\{\hat{\eta}_1\} + \exp\{\hat{\eta}_2\}}$$

$$p(\pi_3 \mid x) = \frac{1}{1 + \exp\{\hat{\eta}_1\} + \exp\{\hat{\eta}_2\}}$$

Given estimates of the values for the population parameters for the model, the first expression can be used to calculate the probability of a new observation with characteristic vector x belonging to class 1, the second expression can be used to calculate the probability of a new observation with characteristic vector x belonging to class 2, and the third expression can be used to calculate the probability of a new observation belonging to class 3. Given the fact that there are only three classes, these probabilities must sum to unity. Then the classification rule is stated as

follows: If faced with the problem of classifying a new observation with characteristic vector x, then classify it as belonging to the class with the highest calculated probability. Extensions to the four-class case and beyond are straightforward.

An important property of MLM is the assumption of independence from irrelevant alternatives (IIA), which could be a major drawback for some practical applications. The property of IIA could be stated as follows: the ratio of the choice of probabilities of any two alternatives is unaffected by the systematic utilities of any other alternatives. In other words, the odds of outcome 1 (say, Path 1) versus outcome 2 (say, Path 2) do not depend on what other outcomes (say, a and b) are available.

2.2 Density Estimation Methods

This is a non-parametric technique that has been studied in statistics by several authors (Silverman, 1986; Wand and Jones, 1995; Hand, 1982; 1997). The main aim of kernel density methods in the discriminant analysis context is to estimate the conditional probability of a class given a set of predictor variables without making parametric assumptions. Its main focus is on estimating the separate distributions of classes.

The univariate kernel estimator can be defined as follows:

$$\hat{f}(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \tag{2.10}$$

where h is the bandwidth of the estimator or the smoothing parameter that controls the degree of smoothing; x_i (i = 1,..., n) comprise a random sample from an unknown density f and the kernel K is taken to be a radially symmetric, unimodal non-negative function that is centred at zero and integrates to one, for instance the multivariate normal density.

For classification tasks the following multivariate product kernel estimator can be used as the basis:

$$\hat{f}(\tilde{x}) = \frac{1}{nh_1...h_d} \sum_{i=1}^{n} \left\{ \prod_{j=1}^{d} K\left(\frac{x_i - x_{ij}}{h_j} \right) \right\} \qquad (2.11)$$

where $\tilde{x} = (x_1,..., x_d)$ and h_j represents the bandwidth of the j-th predictor variable. Here, the same univariate kernel is used in each dimension, but with a different smoothing parameter. The data x_{ij} are collected in an n x d matrix.

Classification is carried out with the kernel density estimates as follows:

Suppose there are k classes, $C_1,..., C_k$, together with a d-dimensional attribute vector $\tilde{x}$. For each class C_j, take only the training data that belongs to class j and estimate the density for the data from that class using $\hat{f}_j(\tilde{x}) = \hat{f}(\tilde{x} \mid C_j)$. Bayes theorem provides a method for classification:

$$\hat{p}(C_j \mid \tilde{x}) = \frac{\hat{f}_j(\tilde{x}) p(C_j)}{\sum_{i=1}^{k} \hat{f}_i(\tilde{x}) p(C_i)} \qquad (2.12)$$

where $p(C_i)$ is the prior probability of class C_j and these are estimated from the data.

One primary advantage of kernel methods is that no training is required to build the model; the training data is the model. Also, the procedure is conceptually quite simple and easily explained. However, Kernels suffer from disadvantages that have kept them from becoming highly used in practice, especially in data mining applications. Since there is no model, they provide no easily understood model summary. As a result, they cannot be easily interpreted. Kernel methods produce a "black box" prediction machine, i.e. in order to make each prediction, the kernel method needs to examine the entire dataset, which could be time consuming for large datasets and requires that the entire dataset be stored in random access memory. Also, kernel classification methods have the weakness of using all the input dimensions as compared to other supervised learning methods (like DTs), which construct a model using only those dimensions that are necessary to discriminate between classes. Other difficulties with kernel density estimation

methods are how to choose the bandwidths $h_1,..., h_d$ for a finite sample size and how to choose the form of kernel. These methods also require very complex computations especially for large datasets. Most seriously, they give very little usable information regarding the structure of the data.

2.3 Nearest Neighbour Methods

One of the most venerable algorithms in machine learning is the nearest neighbour (NN). NN methods are sometimes referred to as memory-based reasoning or instance-based learning (IBL) or case-based learning (CBL) techniques and have been used for classification tasks. They essentially work by assigning to an unclassified sample point the classification of the nearest of a set of previously classified points.

The entire training set is stored in the memory. To classify a new instance, the Euclidean distance (possibly weighted) is computed between the instance and each stored training instance and the new instance is assigned the class of the nearest neighbouring instance. More generally, these *k*-nearest neighbours (*k*-NNs) are computed, and the new instance is assigned the class that is most frequent among the *k* neighbours. IBL's have three defining general characteristics: a similarity function (how close together the two instances are), a "typical instance" selection function (which instances to keep as examples), and a classification function (deciding how a new case relates to the learned cases) (Aha *et al.*, 1991).

Consider a set of *n* pairs is $(x_1, C_1),...,(x_n, C_n)$, where x_i's take values in the metric space X upon which is defined a metric d, and the C_i's take values in the set {1, 2, ..., *K*}. A new measurement x is observed, and it is desired to estimate C by utilising the information contained in the set of correctly classified points. Cover and Hart (1967) call $x'_n \in \{x_1,..., x_n\}$ a NN to x if $\min d(x_i, x) = d(x'_n, x)$ $i = 1, 2,..., n$. The NN rule decides that x belongs to the category C'_n of its nearest neighbour x'_n. A mistake is made if $C'_n \neq C$. Notice that only classification of the NN is utilised by this, simplest, nearest neighbours rule. The remaining *n-1* classifications C_i are ignored.

A further non-parametric procedure of this form is the k-nearest neighbour (k-NN) approach. For each element in the test set, the k-NN approach finds the k closest observations in the training set. To classify an unknown pattern, a collection of the k nearest points are looked at and a "majority voting" mechanism to select between them is used, instead of looking at the single nearest point and classifying according to that with ties broken at random. If there are ties for the k^{th} nearest observations, all candidates are included in the vote.

Despite its simplicity and making minimal assumptions about the format of the underlying distribution, the nearest neighbour (NN) classifier has the advantage of being able to learn from a small set of examples. It also has the ability to incrementally add new information at run time (i.e., nearest neighbor methods do not need any computation up-front. All the computation occurs at run time when a classifier is asked to produce a class label for a new unknown instance.). Lastly, NNCs can give competitive performance with more modern methods such as DTs or neural networks.

One difficulty with k-NN methods is the choice of the distance metric k that is unknown for finite n. In fact, the distance function could be Euclidean, Manhattan, Mahalanobis, Pearson, etc. Also, the NNC has the limitation of being slow in runtime performance and requires very large memory. This is as a result of storing and recalling of the training sample each time a new case is classified. The classifier cannot be constructed beforehand. Also, the computation of nearest neighbour distances is expensive in high dimensions. Futhermore, there are no natural or simple ways to handle discrete variables and missing data. Notice that some of these limitations are common for both kernel density estimation and k-th nearest neighbour methods.

2.4 Support Vector Machines

Support Vector Machines (SVMs) are pattern classifiers that can be expressed in the form of hyperplanes to discriminate positive instances from negative instances pioneered by (Vapkin, 1995, Burges, 1998). The principal goal of the SVM approach is to fix the computational problem of predicting with kernels (Breiman *et al.*, 1984),

as discussed in Section 2.1.2. The basic idea of SVMs is to determine a classifier or regression machine which minimizes the empirical risk (i.e., the training set error) and the confidence interval (which corresponds to the generalisation or test set error). In other words, the idea is to fix the empirical risk associated with architecture and then use a method to minimize the generalisation error. Motivated by statistical learning theory, SVMs have successfully been applied to numerical tasks, including classification. They can perform both binary classification (pattern recognition) and real valued function approximation (regression estimation) tasks.

The key understanding of SVMs is to see how they introduce optimal hyperplanes to separate classes of data in the classifiers. Intuitively, given a set of points which belong to either of two classes, a linear SVM finds the hyperplane leaving the largest possible fraction of points of the same class on the same side, while maximising the distance of either class from the hyperplane. The hyperplane is determined by a subset of the points of the two classes, named support vectors, and has a number of interesting theoretical properties.

SVMs have the strength of being highly flexible as regards the kind of problem which may be tackled. They can be used to classify both linearly and non-linearly separable data. Another benefit of SVMs is that the complexity of the resulting classifier is characterised by the number of support vectors rather than the dimensionality of the transformed space. As a result, SVMs tend to be less prone to problems of overfitting than other methods.

SVMs share the some weaknesses of ordinary kernel methods. They are a black-box procedure with little interpretive value. Outliers in the data can distort the boundaries. Therefore, before using SVMs, data should be cleaned to clarify that outlying points are supplying genuine information rather than false measurement and so on. As with all kernel methods, the choice of the kernel when classifying nonlinear separable data, which is not only problem specific rather than adhering to global rules but could also be very sensitive to the performance is another weakness of SVMs. SVMs can take longer to train and they are more suited to problems with

numerical data rather than categorical. Also, determining a suitable learning algorithm for minimizing empirical risk may be quite difficult.

2.5 Artificial Neural Networks

Networks that mimic the way the brain works is just one of the many claims and attractions of Artificial Neural Networks (ANNs). ANNs are inspired by the strong interconnectedness of the human brain. The network learns in the training phase by having its weights (strengths) adjusted such that the actual network output becomes more similar to the desired output.

An ANN is a set of one or more layers of nodes that are interconnected by directed weighted links. Each link has an associated weight (strength). Links with positive weights are called excitatory links and links with negative weights are called inhibitory links.

Artificial Neural Networks, usually nonparametric approaches, are represented by connections between a very large number of simple computing processors or elements (neurons), have been used for a variety of classification and regression problems. These include pattern and speech recognition, credit card fraud detection, chemical process control, robotics, and so on (Winston, 1992; Ripley, 1992; 1994; 1996; Patterson, 1996). There are many types of ANNs, but we shall concentrate briefly on single unit perceptrons and multi layer perceptrons also known as "backpropagation networks".

The backpropagation learning algorithm performs a hill-climbing search procedure on the weight space described above or a (noisy or stochastic) gradient descent numerical method whereby an error function is minimised. At each iteration, each weight is adjusted proportionally to its effect on the error. One cycles through the training set and on each example changes each weight proportionally to its effect on lowering the error. One may compute the error gradient using the chain rule and the information propagates backwards through the network through the interconnections, which accounts for the procedure's name.

There are two stages associated with the backpropagation method: training and classification. The ANN is trained by supplying it with a large number of numerical observations or the patterns to be learned (input data pattern) whose corresponding classifications (target values or desired output) are known. During training, the final sum-of-squares error over the validation data for the network is calculated. The selection of the optimum number of hidden nodes is made on the basis of this error value. The question of how to choose the structure of the network is beyond the scope of this thesis and is a current research issue in neural networks. Once the network is trained, a new object is classified by sending its attribute values to the input nodes of the network, applying the weights to those values, and computing the values of the output units or output unit activations. The assigned class is that with the largest output unit activation.

Neural network-based approaches are usually non-parametric even though statistical information could be possibly incorporated to improve their performance (for example, speed of convergence). Even though ANNs explore a rich hypothesis space with a "bias" that seems to work well in many domains, and they also offer most advanced classification power because of their capability of implementing complex mappings due to their multi-layer structure and the use of non-linear threshold functions, a number of difficulties with them have become evident with experimentation.

Despite their strengths, ANNs have the weakness of lacking explicitness, i.e., the process is most of the time unexplained; they also take a long time to learn to recognise underlying patterns in data. ANNs classifiers are like "black boxes" – they are very hard to interpret. There is also the difficulty of choosing the network topology (the number of hidden units and initial weights to choose). As in LR, categorical predictors are treated via dummy vectors. Furthermore, ANNs are highly redundant and overparameterized because of the many examples they use, thus, many parameters need to be estimated. This not only slows down the learning process but makes the method susceptible to noise and overfitting.

2.6 A 'Naïve' Bayes Classifier

One role for Bayesian methods is to provide practical learning algorithms such as naïve Bayes learning, Bayesian belief network learning, and combining prior knowledge (prior probabilities) with observed data (Cooper, 1990; Cooper and Herskovitz, 1992; Buntine, 1994; Jensen, 1996; Heckerman *et al.*, 1994; Heckerman, 1996).

The purpose of learning or training is to make predictions for future cases in which only the inputs to the network are known. The result of conventional network training is a single set of weights that can be used to make such predictions. In contrast, the result of Bayesian network training is a posterior distribution over network weights. If the inputs of the network are set to the values for some new case, the posterior distribution over network weights will give rise to a distribution over the outputs of the network, which is known as the predictive distribution for this new case. If a single-valued prediction is needed, one might use the mean of the predictive distribution, but the full predictive distribution also tells you how uncertain this prediction is. Bayesian learning algorithms use probability theory as an approach to concept classification. Bayesian classifiers produce probabilities for (possibly multiple) class assignments, rather than a single definite classification.

Bayesian learning should not be confused with the "Bayes optimal classifier." Also, Bayesian learning should not be confused with the "naïve" or "idiot's" Bayes classifier (see Figure 2.1), which assumes that the inputs (or attributes) are conditionally mutually independent given the target class. The naïve Bayes classifier is usually applied with categorical inputs, and the distribution of each input is estimated by the proportions in the training set; hence the naïve Bayes classifier (NBC) is a frequentist method.

The NBC is perhaps the simplest and most widely studied probabilistic learning method. It learns from the training data, the conditional probability of each attribute A_i given the class label C (Duda and Hart, 1973; Ripley, 1992; Mitchell, 1997). The strong major assumption is that all attributes A_i are independent given the value of the class C. Classification is therefore done applying Bayes rule to

compute the probability of C given $A_1,..., A_n$ and then predicting the class with the highest posterior probability. The probability of a class value C_i given an instance $X = \{A_1,..., A_n\}$ for n observations is given by:

$$\begin{aligned} p(C_i|X) &= p(X|C_i).p(C_i)/p(X) \\ &\alpha\, p(A_1,..., A_n|C_i).p(C_i) \\ &= \prod_{j=1}^{n} p(A_j|C_i).p(C_i) \end{aligned} \tag{2.13}$$

The assumption of conditional independence of a collection of random variables is very important for the above result. Otherwise, it would be impossible to estimate all the parameters without such an assumption. This is a fairly strong assumption that is often not applicable. However, bias in estimating probabilities may not make a difference in practice – it is the order of the probabilities, not the exact values that determine the probabilities.

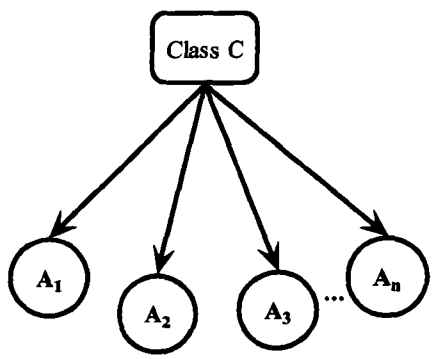

Fig. 2.1. A "naïve" Bayes classifier; the arrows represent causal influence

When the strong attribute independence assumption is violated, the performance of the NBC might be poor. However, Domingos and Pazzani (1996) argue that the NBC is still optimal even when the independence assumption is violated as long as the ranks for the conditional probabilities of classes given and instance are correct. A few techniques (Kononenko, 1991; Langley and Sage, 1994; Zheng and Webb, 1997) have been developed to improve the performance of the NBC. However, more work still has to be done on this independence assumption violation.

The NBC was also found to be inherently robust with respect to noise, and scales well to domains that involve irrelevant features and large datasets (Langley, 1993; Sing, 1997). The "naïve" classifier has also been shown to give remarkably high accuracies in many natural domains (Langley *et al.*, 1992). However, despite its strengths (of quick learning and low computational overhead), this approach is limited to learning classes that can be separated by only a single decision boundary (Langley and Sage. 1994) as opposed to other learning algorithms (like DTs) that recursively partition the instance space into sub-regions. For each partitioning there is a corresponding DT. Thus, the classes are separated by more than one decision boundary. The NBC is also limited in expressiveness in that it can only create linear frontiers (Domingos and Pazzani, 1996). Also, even with many training instances and no noise, the NBC does not approach 100% (maximal) accuracy on some problems. Langley (1993) proposed the use of 'recursive Bayesian classifiers' to address this limitation. NBC suffers from domains where the feature variables are correlated given the class variable. Also, the NBC is restricted to categorical variables even though some work on using continuous variables has been made (Gammerman *et al.*, 1995).

2.7 Rule Induction

This is sometimes called affinity grouping. The systems learn a set of condition-action rules from data (such as: "IF item *A* is part of an event, THEN x percent of the time, item B is part of products, training and support"). Disjunction Normal Form is a logical formula consisting of a disjunction of conjunctions where no conjunction contains a disjunction. For example, the Disjunction Normal Formula of (A or B) and C is (A and C) or (B and C). This is the most common representation used for rules induced by such algorithms. Examples of rule learning systems are AQ15 (Michalski *et al.*, 1986), CN2 (Clark and Niblett, 1989), GREEDY3 (Pagallo and Haussler, 1990), PVM (Weiss *et al.*, 1990), RIPPER (Cohen, 1995) and many more. These algorithms induce rules for each class and instances directly whereby each rule covers a subset of the positive instances, and few or no negative ones, then "separating out" the newly covered examples and starting again on the remainder.

The rule is composed of a consequent (predicted class) and an antecedent (a condition involving a single attribute).

Learning algorithms in the rule-induction framework usually select the "best" rule by a greedy search through a space of rule sets and utilising the evaluation function to select attributes for incorporating in the knowledge structure. The choice of the evaluation function is of great importance to that algorithm's performance. The AQ algorithm (Michalski *et al.* 1986) uses the accuracy of the rule on the training set given in a formula as:

$$E(i_+, i_-) = \frac{i_+}{i_+ + i_-} \tag{2.14}$$

where i_+ is the number of positive instances that satisfy the rule and i_- is the number of negative instances that satisfy the rule. Given the rule, E should increase with i_+ and decrease with i_-. The AQ and CN2 systems originally used the entropy rule (Quinlan, 1986) as its search heuristic. The entropy measure not only prefers rules which cover examples of only one class, it was not able to generate an ordered set of rules. To overcome this problem a Laplace error estimate approach is used by Niblett (1987):

$$E(i_+, i_-) = \frac{i_+ + 1}{i_+ + i_- + c} \tag{2.15}$$

where c is the number of classes.

When generating a rule list, the predicted class for a rule is simply the class with the most covered instances in it. Classification of a new instance is performed by matching each rule with that particular instance, and selecting those it satisfies. If there is only one such rule, its class is assigned to the instance. If there are none a "default rule" is used, i.e., assigning the instance to the class that occurs most frequently in the entire training set, or among those instances not covered by any rule. Finally, if more than one rule covers the example one strategy is to order rules into a "decision list", and select only the first rule that fires (Rumelhart *et al.*, 1986). The other strategy is to let different rules vote, and select the class with most votes (Clark and Boswell, 1991).

Rule Learners' strength lies in their comprehensibility since they have a very low storage requirement and can partition subspaces of the instance space. Rules are also easy for people to understand and modify, and are therefore good for providing insights about regularities in the data. Since there are a lot of variables to tune, rule induction is slow in training. Rule learning systems use search procedures that could become intractable if the space of variables gets too large. The main difference between rule learners and DTs is that even though they both partition a space; rule learners do not do it recursively in the manner of trees.

2.8 Decision Trees

DT induction is one of the simplest and yet most successful forms of supervised learning algorithm. It has been extensively pursued and studied in many areas such as statistics (Kass, 1980; Clark and Pregibon, 1993; Breiman *et al*, 1984; Jordan, 1994; Vach, 1995) and machine learning (Quinlan, 1979; 1983; 1986; 1993) for the purposes of classification and prediction.

DT classifiers have four major objectives. According to Safavian and Landgrebe (1991), these are: 1) to classify correctly as much of the training sample as possible; 2) generalise beyond the training sample so that unseen samples could be classified with as high accuracy as possible; 3) be easy to update as more training samples become available (i.e., be incremental – see sub-section 2.2.5); 4) and have as simple a structure as possible. Objective 1) is actually highly debatable and to some extent conflicts with objective 2). Also, not all tree classifiers are concerned with objective 3).

DTs are non-parametric (no assumptions about the data are made) and a useful means of representing the logic embodied in software routines. A DT takes as input a case or example described by a set of attribute values, and outputs a Boolean or multi-valued "decision". For the purpose of this thesis, we shall stick to the Boolean case. A brief overview about Boolean expressions is given below.

A Boolean value is either True or False. A Boolean variable may only take a Boolean value. The negation of a Boolean value is the other Boolean value (for example, the

negation of True is False). The symbol often used to denote the negator, *Not*, is '¬' (for example, ¬ True ≡ False). There are sixteen Boolean junctors, binary operators for joining two Boolean variables. The most commonly used Boolean junctors are *And* and *Or*. The *And* junctor is often denoted by the symbol '∧' and is known as the conjunctor. The *Or* junctor is often denoted by the symbol '∨' and is known as the disjunctor. Each of the following is a Boolean expression: a Boolean value, a Boolean variable, a negation of a Boolean expression, the conjunction of two Boolean expressions, the disjunction of two Boolean expressions. Each occurrence of a variable in an expression is known as a literal.

From a Boolean expression we can construct a DT, which actually looks more like an upside down tree of nodes, as follows; each negator, junctor and literal is converted to a node, nodes which become parents of other nodes are placed higher than their children and connected by a line to each child, the operand of a negator is made the right child of that negator, the left and right operands of a junctor become left and right children of that junctor, and parentheses, after being used to specify the structure of the expression, are redundant and are implied by the structure of the tree rather than being represented explicitly.

The power of this approach comes from its ability to split the instance space into subspaces where each subspace is fitted with different models. Other distinctive advantages of Boolean expressions are their time complexity and space complexity, i.e., as the input size as Boolean functions grow linearly, computer time and storage requirements of algorithms to represent and manipulate DTs do not necessarily grow exponentially (as is the case for traditional representation techniques). Thus, both the memory space to store DT and the time of the learning and using phase are decreased (Murphy and McCraw, 1991).

Boolean attributes have also been used to incorporate continuous-valued attributes so that they compete with other discrete-valued candidate attributes available for growing the DT. In particular, for an attribute that is continuous-valued, the DT algorithm can dynamically create a new Boolean attribute A_c that is true if $A < c$ and false otherwise. Further, information-based attribute selection measures used in

the induction of DTs are all biased in favour of attributes that have many possible values (White and Liu, 1994; White, 2001). The use of Boolean attributes is one strategy for handling multi-valued attributes and as a result overcomes this problem, i.e., attributes with fewer levels and those with many levels are all selected on a competitive basis.

We are given a set of instances. Each instance has the same structure consisting of a number of attribute/value pairs. One of these attributes represents the category of the instance. The problem is to determine a DT that on the basis of answers to questions about the non-category attributes predicts correctly the value of the category attribute. Sometimes the category takes only the values {true, false} or {success, failure}, or something equivalent. In any case one of its values will mean failure.

Graph representation of Boolean expressions are represented by binary DTs i.e., tree-based algorithms allow the creation of a family of Boolean expressions for one to be able to draw the corresponding binary partitions and as a result binary DTs. Therefore, from now on in this thesis, we shall be using the terminology binary DT to mean graph representation of Boolean expressions.

Consider the following artificial example taken from Quinlan (1979), where the categorical attribute specifies whether to play or not to play a game of golf. The attributes are:

Table 2.1 Attribute variables and values

ATTRIBUTE	POSSIBLE VALUES
Outlook	sunny, rain
Temperature (F)	continuous
Humidity	continuous
Windy	true, false

For this example, the outlook attribute has only two possible values compared to the original example by Quinlan which has three values (i.e. sunny, outcast and rain). The training data is shown in Table 2.2.

Table 2.2 Artificial outlook dataset

OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	CLASS
Sunny	85	85	False	Don't play
Sunny	80	90	True	Don't play
Rain	83	78	False	Play
Rain	70	96	False	Play
Rain	68	80	False	Play
Rain	65	70	True	Don't play
Rain	64	65	True	Play
Sunny	72	95	False	Don't play
Sunny	69	70	False	Play
Rain	75	80	False	Play
Sunny	75	70	True	Play
Rain	72	90	True	Play
Rain	81	75	False	Play
Rain	71	80	True	Don't play

The training data is used to build the model (a DT) and a corresponding hierarchical partitioning shown in Figures 2.2 (a) and 2.2 (b), respectively.

The DT has classes *play* and *don't play*. This is a classifier in the form of a tree, a structure that is either: a leaf (rectangular box), indicating a class, or a decision node (ovals) that specifies some test to be carried out on a single attribute value, with one branch and subtree for each possible outcome of the test. Notice that Test 2 is a test on the real-valued attribute *Humidity, i.e., Humidity* > cut-off point. The

other tests use the nominal attributes. Further note that the attribute variable temperature has been not utilised in the DT even though we have it as one of the attributes in the training data. It is also easy to see that as the depth of the tree increases, the resulting partitioning becomes more and more complex.

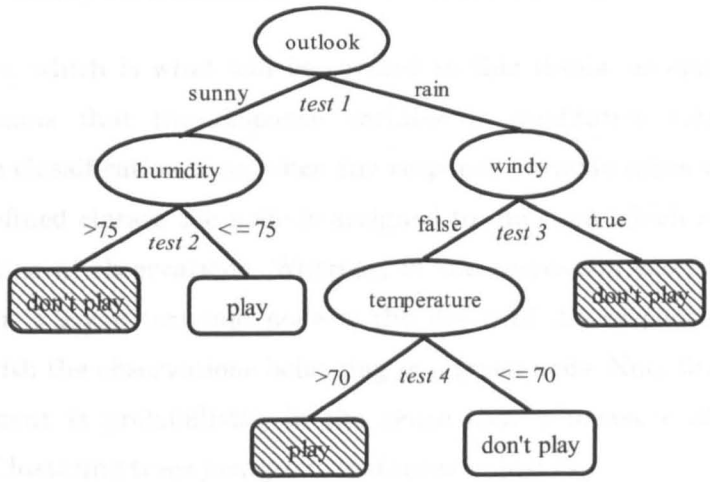

Fig. 2.2. **(a)** Example of a binary axis - parallel decision tree for a four-dimensional feature space and 2 classes for deciding whether or not to play golf. Ovals represent internal nodes; rectangles represent leaf nodes or terminal nodes.

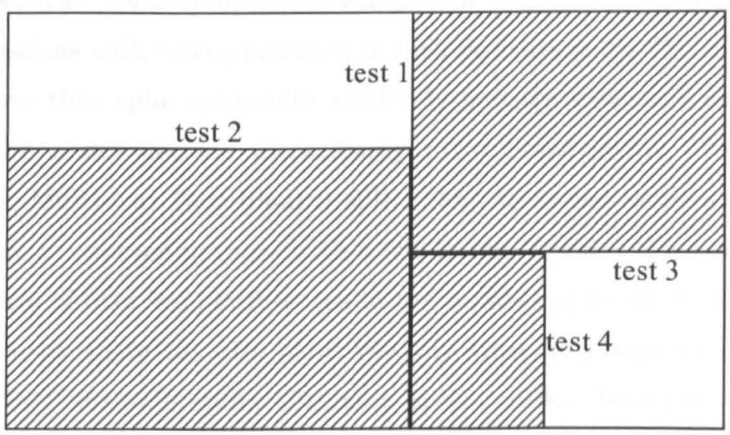

Fig. 2.2. **(b)** Hierarchical partitioning of the two-dimensional feature space induced by the decision tree of Figure 2.5 (a)

The prediction problem is handled by partitioning classifiers. These classifiers split the space of possible observations into partitions. For example, when a person needs to make a decision whether to play golf or not based on several factors (e.g. weather conditions), a classification tree can help identify which factors to consider and how each factor has historically been associated with different outcomes of the decision.

A classification tree, which is what will be covered in this thesis, as opposed to a regression tree means that the response variable is qualitative rather than quantitative. In the classification case, when the response variable takes value in a set of previously defined classes the node is assigned to the class which represents the highest proportion of observations. Whereas, in the regression case, the value assigned to cases in a given terminal node is the mean of the response variable values associated with the observations belonging to a given node. Note that in both cases, this assignment is probabilistic, in the sense that a measure of error is associated with it. Clustering trees just group instances in leaves.

On another note, the DTs we shall be looking at in our research are binary (Yun and Fu, 1983; Breiman *et al.*, 1984; Clark and Pregibon, 1993; Venables and Ripley, 1994; Therneau and Atkinson, 1997) as opposed to non-binary. Binary trees use strictly binary, or two-way, splits that divide each parent node into exactly two child nodes by posing questions with yes/no answers at each decision node. The algorithm searches for questions that split nodes into relatively homogenous child nodes. As the tree evolves, the nodes become increasingly more homogenous, identifying important segments. Multi-way (non-binary) splits tend to favour attributes with many levels or values as opposed to those with a few levels. The use of binary splits overcomes the problem of bias from differences in the number of levels of attributes (White, 2001). Multi-way splits can also paint visually appealing trees but can bog models down with less accurate splits (Breiman *et al.*, 1984). However, a large amount of work has also been done on non-binary classification trees, see (Kass, 1980; Biggs *et al.*, 1991; Quinlan, 1986; 1993; Kerpta, 1996).

A tree is characterised by an ordered set of nodes. Each of the internal nodes in the tree represents a partition of one or more features. A DT induces a hierarchical

partitioning over decision space. The root is the top node (refer to Test 1 in Figure 2.2 (a)) and the training sample is passed down the tree, with decisions or predicted probabilities of the classes being made at each node until the terminal node (or leaf node) is reached. Each leaf contains the label of a classification. Each node, except the leaves, is a parent node, which is linked to its children. Basically, at each node the best variable is chosen, where the best means most homogenous in terms of child nodes. Since the data is recursively partitioned, each node contains a subset of the initial population with the root containing the whole training sample and the other nodes contain sub-groups of the training sample.

DTs use a top down recursive partitioning process, i.e., a divide-and-conquer strategy or targeted stratification that "greedily" searches (a "look-ahead" principle like forward stepwise regression) over all the variables to produce a sequence of optimal splits, so that a large tree is grown. The core algorithm that builds a DT from data is summarised in Figure 2.3.

Input: A set of training examples. Stopping and splitting rules.

Output: A DT with class counts at the leaves

1. Find out how many of the training examples belong in each class at the current terminal node (leaf)
2. If all training examples belong to one class or if some *stopping rule* applies, the leaf is labelled with that class.
3. Otherwise,
   a) select a test using a *splitting rule*, based on one attribute, with mutually exclusive outcomes;
   b) divide the training set into subsets of examples, each corresponding to one outcome, and
   c) apply the same procedure to each subset

Fig. 2.3. The decision tree induction algorithm

Learning systems that are based on this approach belong to the Top Down Induction of Decision Trees (TDIDT). If the observations from the classes overlap (a typical

problem involving noise), we over-fit the training sample at hand and try and stop the tree from growing either by stopping growing the tree or by pruning the tree after constructing it. A single tree is then selected according to a model choice criterion (e.g. cost-complexity pruning, cross validation error-based pruning or even multiple tests of whether two adjoining nodes should be collapsed into a single node). Details of all these aspects will be developed in the remainder of this section. In top-down approaches, the process of tree development involves selection of the next feature; specification of the threshold (cut-off point) on that feature; determining which nodes should be terminal nodes and assignment of each terminal node to a class label. This can be summarised in three important stages: a splitting rule, a stopping rule, and a pruning rule.

Of the above tasks, the class assignment is by far the easiest. To minimise the misclassification rate, the terminal nodes are assigned to the classes which have the highest probabilities. These probabilities are usually estimated by the ratio of the samples from each class at that specific terminal node to the total number of samples at that specific terminal node. The label of the class that has the most samples at that terminal node is assigned to the terminal node.

2.8.1 Splitting Rules

The problem of designing a truly optimal DT seems to be very difficult. In fact, Hyafil and Rivest (1976) and later Rounds (1980) have shown that finding a minimal DT consistent with a set of data is a NP-complete problem. A problem is called NP (nondeterministic polynomial) if its solution (if one exists) can be guessed and verified in polynomial time; nondeterministic means that no particular rule is followed to make the guess. Also, Hyafil and Rivest (1996) conjecture that no sufficient tree learning algorithm exists and thereby supply motivation for finding efficient heuristics for constructing near-optimal DTs. Furthermore, Dasarathy (1980), and Cohen and Martin (1997) show that even the problem of identifying the root node in an optimal strategy is NP-hard. The problem of building optimal trees is looked at by Murthy *et al.* (1994) who proved that for most cases, construction of storage optimal trees is NP-complete

Tree learning algorithms use heuristics which perform a one-step look-ahead search which works as a splitting rule to create subsets of data obtained. The system chooses the test that maximises or minimises some heuristic function over the subsets that are relatively "pure" in one class, and chooses the split that has the fewest errors. This is sometimes referred to as the evaluation or selection test and is guided by a measure of "goodness of split".

To construct a tree it is necessary to assess the quality of a given test, based on the attribute values, and the resulting partition. A splitting rule works by finding the variable (or test) which is most discriminatory and partitioning the data with respect to that variable. The variable that gives the strongest association with class is selected to branch on.

The binary tree partitions each non-terminal node into two branches in the following way. Each internal node n has an associated splitting rule and is labelled with one predictor attribute X_n called the splitting attribute. Each internal node has a predicate q_n, called the splitting predicate associated with it. For an ordered attribute at node X_n, q_n is of the form $X_n \leq x_n$ or $X_n > x_n$ (where $x_n \in domain(X_n)$; x_n is the split point at node n) was used. For a nominal attribute, q_n is of the form $X_n \in Y_n$ or $X_n \notin Y_n$ (where $Y_n \subset domain(X_n)$; Y_n is called the splitting subset at node n) was used. In other words, a split for numeric attributes is carried out by: sorting the training instances based on the values of the attribute being considered; sorting out the values of the numeric attribute and dividing the set into two subsets using some interval; and finally selecting the midpoint for each interval as the split point. A split for nominal attributes is carried out by considering all possible partitions of that attribute into two non-empty categories. An ordered attribute with M distinct values yields M-1 distinct splits. For a nominal attribute with L distinct values there are $2^{L-1}-1$ distinct splits to consider.

Several measures of node homogeneity or impurity have been proposed, but the two most commonly used (based on the information gain) are the *entropy impurity function* and the *Gini index of impurity*. These two are also based on probabilities

estimated from the training examples. Most of the measures are based on information theory, distance, dependence and other areas. Some of them have been reviewed and compared by Mingers (1989b) who surprisingly concluded that the choice of measure affects only the tree size but not its accuracy. According to Mingers the accuracy remains the same even when the attributes are randomly selected. Mingers' claim was questioned by Buntine and Niblett (1992) and further by Liu and White (1994) who used more experiments to justify why they refuted the claim.

We shall now look at some impurity measures.

2.8.1.1 Information Gain Measure

This measure originates from information theory (Shannon, 1948a; 1948b), and is computed by the following formula.

Using Quinlan's (1986) notation, the information needed for distinguishing k classes of randomly drawn members of training set T is given by:

$$\text{Entropy(T)} = -\sum_{i=1}^{k} p_i \log_2 p_i \qquad (2.16)$$

where $p_i = \frac{N_i}{N}$, where N is the total number of cases at a particular node and N_i is the number of cases in class i. In other words, p_i is the proportion of T belonging to class i. These probabilities are estimated from the training examples. The logarithm is base 2 because entropy is the measure of the expected encoding length measured in bits. Note also that the maximum possible entropy is $\log_2 k$, obtained when the classes are equiprobable.

The information needed for distinguishing classes of randomly drawn members of T after the value of attribute A has been obtained is defined as:

$$\text{Entropy(A)} = \sum_{j \in \text{Values(A)}} \frac{|T_j|}{|T|} \text{Entropy}(T_j) \qquad (2.17)$$

where *Values* (*A*) is the set of all possible values of the predicate for attribute *A*, and T_j is the subset of *T* for which attribute *A* has value *j*, and the entropy of the partitioned data is calculated by weighting the entropy of each partition by its size relative to the original set. Here, $|T|$ denotes the number of elements of set *T*. This is the information needed to classify items given knowledge of the attribute value or simply the sum of the entropies of each subset T_j, weighted by the fraction of instances $|T_j| / |T|$ that belong to T_j.

The information measure ("goodness of split") of an attribute is defined to be the gain in information brought about by the knowledge of the attribute:

$$\text{Gain}(T, A) = \text{Entropy}(T) - \text{Entropy}(A) \qquad (2.18)$$

Gain(T, A) can also be defined as the expected reduction in entropy caused by partitioning on this attribute. The most 'informative' attribute is the one that minimises Entropy(A), i.e., it maximises *Gain*. The value of Gain(T, A) is the number of bits saved when encoding the target attribute value of an arbitrary member of T, by knowing the value of attribute *A*. The corresponding heuristic selects the attribute that results in the maximum gain for that step. The process of selecting a new attribute and partitioning the training instances is repeated for each non-terminal descendant node, this time using only the training instances associated with that node.

2.8.1.2 Gain Ratio Measure

The information gain heuristic has been shown to favour many-valued attributes over one with few, which tend to discriminate better among classes because they have more values (Kononenko *et al.*, 1984; Quinlan, 1986; White and Liu, 1994; Ripley, 1996, Luzowski, 1996). For example, if we have an attribute *D* that has a distinct value for each instance, then *E*(*D*) is 0, thus *Gain* (*D*) is maximal. To compensate for this a normalisation procedure is proposed by Quinlan (1993). This measure incorporates the information measure (entropy) of an attribute, Gain (A), with the information value of the attribute, IV(A), as follows:

$$GR(A) = \frac{Gain(A)}{IV(A)} \tag{2.19}$$

where, $IV(A) = \sum_{j=1}^{J_k} p_j \log_2 p_j$.

2.8.1.3 The GINI Index of Impurity

This measure, which is similar to the information gain measure but based on a different function, was introduced by Breiman *et al.* (1984) and, at node *t*, it is computed as follows:

$$i(t) = \sum_i \sum_{j \neq i} p_i p_j = \sum_i \sum_j p_i p_j - \sum_i p_i^2 = \left(\sum_i p_i \right)^2 - \sum_i p_i^2 = 1 - \sum_i p_i^2 \tag{2.20}$$

where $p_i = p(c_i \mid t)$ is estimated from the relative frequencies of the classes in the same way as in the previous section. The GINI index has also been described as the estimated probability of the misclassification error or as a measure of how different the members of a set of probabilities are from each other. The decrease in impurity is the class impurity minus the weighted average of the implied descendant nodes' impurity given by:

$$\Delta i(s_p, t) = i(t) - p_{t_l} i(t_l) - p_{t_r} i(t_r)$$

where $i(t_l)$ and $i(t_r)$ are the impurity measures at sub-nodes t_l and t_r, respectively; p_{t_r} and p_{t_l} are the proportions of the instances that are sent by the split s_p of a node t to the right t_r and left t_l nodes, respectively. The idea in CART is to calculate $\Delta i(s_p, t)$ for each split $s_p \in Q_t$ is the set of all possible splits at node t and then select the best split s_p^* such that

$$\Delta i(s_p^*, t) = \max_{S_p \in Q_t} \Delta i(s_p, t).$$

However, since the decrease in impurity is essentially linear among the instances, it has the defect of favouring the more extreme one from two competing splits. For example if there were two competing splits, one separating the data into 80% and 20% groups of purity and the other split into 70% and 30%, we would prefer the

former because it better sets things up for the next split. Also, the Gini criterion has a difficulty when there are a relatively large number of classes. Breiman *et al.* (1984) suggested the towing rule (*twoing* and *ordered twoing*) as a remedy. The twoing criterion requires the selection at every node be divided in two superclasses, i.e., both methods are designed as a binary partition and group the classes from the original set of classes (c) into two mutual subclasses c_1 and c_2 (c - c_1). Then, as if it were a two class problem, compute the decrease in impurity for c_1 ($\Delta i(s_p, t, c_1)$), given that $\Delta i(s_p, t)$ depends on the selection of c_1. A split s_{Pc_1} which maximises $\Delta i(s_p, t, c_1)$ and the superclass c_1^{sc} which maximises $\Delta i(s_{Pc_1}, t, c_1)$ are finally found. The criterion attempts to group together classes that are similar in some characteristics near the top of the tree and attempts to isolate single classes near the bottom of the tree. It is an intuitive criterion that attempts to inform the user of class similarities (Breiman *et al.*, 1984).

The twoing criterion for any node t and split s into left node, t_L, and right node, t_R is defined by

$$i(t) = \frac{p_{t_l} p_{t_r}}{4} \left(\sum_j | p(c_j \mid t_l) - p(c_j \mid t_r) | \right)^2 \qquad (2.21)$$

The split that maximises the twoing criterion at a node is determined as the best split for this node. For a discrete attribute twoing investigates each possible combination of values resulting in two superclasses. For continuous attributes the data is sorted and the midpoint between each data sample is used as the sample split. Once the twoing criterion is maximised, the split defined by this function is applied. Ordered twoing restricts the classes in each subset to be ordered and requires, for each possible ordered superset class, the evaluation of the numbers in the "super" class.

Towing is used when there are more than two classes (if there are two classes the twoing criterion is the same as the Gini criterion). The twoing criterion has the significant advantage of giving the user additional insight concerning the structure of the data from the output (i.e. by informing the user of class similarities). Twoing also seems desirable with a large number of classes, but has a drawback in terms of

computational efficiency. For example, for J classes, there are 2^{J-1} distinct divisions into two groups.

2.8.1.4 Chi-Square Statistic

This is a traditional statistic for measuring association between two variables for categorical data in a contingency table. (Haussler, 1988; Mingers, 1989b; Mitchell, 1997) employ the χ^2 statistic to evaluate the attributes and then select the most important attribute.

$$\chi^2 = \sum_i \sum_j \frac{(E_{ij} - x_{ij})^2}{E_{ij}} \tag{2.22}$$

where x_{ij} is the observed number of cases with value X_i in class Y_j and $E_{ij}=x_i x_j/N$ is the expected number for each observation. The resulting statistic is distributed approximately as a chi-square distribution, with larger values indicating higher or greater association between the attributes and the class variable. In other words the statistical test is used to decide whether expanding a node is likely to improve performance over entire instance distribution, or only on current sample of training data.

2.8.1.5 Normalised Information Gain

This attribute selection measure was proposed by Lopez de Màntaras (1991), and is related to Quinlan's information gain measure. The main difference is the former's reliance on a distance measure on any partitions P_A, P_B and P_C of X. The measure is given by:

$$d(P_A, P_B) = I(P_B/P_A) + I(P_A/P_B) \tag{2.23}$$

where P_A and P_B are partitions of A and B; $I(P_B/P_A)$ and $I(P_A/P_B) = I(P_A,P_B)/P_B$ are the average information gain in partition P_B given P_A and P_A given P_B, respectively. For example, $I(P_B/P_A) = I(P_B \cap P_A) - I(P_A)$. The measure $d(P_A,P_B)$ should satisfy the following conditions:

(i) $d(P_A, P_B) \geq 0$, and the equality holds *iff* $P_A = P_B$

(ii) $d(P_A, P_B) = d(P_B, P_A)$

(iii) $d(P_A, P_B) + d(P_A, P_C) \geq d(P_B, P_C)$.

The normalisation is given by the following distance (Lopez de Màntaras, 1991):

$$d(P_A, P_B) = \frac{d(P_A, P_B)}{I(P_A \cap P_B)} \tag{2.24}$$

For proofs the reader is referred to Lopez de Màntaras (1991).

Màntaras re-writes Quinlan's information for distinguishing classes and the expected information content of the tree with attribute A as root as $I(P_C)$ and $I(P_C \mid P_V)$, respectively. Therefore, the information gain measure can now be defined as:

$$\text{Gain}(A) = I(P_C) - I(P_C \mid P_V)$$

So,

$$d(P_A, P_B) = I(P_A) + I(P_B) - 2\text{Gain}(A). \tag{2.25}$$

The node that is selected is the one whose corresponding partition is the closest (in terms of some distance) to the correct partition of the subset of examples in the node. The idea behind this method is very simple. The different values of each attribute generate a partition of the training examples. That is, if an attribute has two values, say, 0 and 1, those training examples that have value 0 for that attribute are in one class of the partition, and those that have value 1 are in the other class. If all those that are in class 0 are of the same target class and those in class 1 are also of the same target class then the distance is 0 and this attribute would therefore be the best one. In general, however, there are examples of different target classes in each of the classes of the partition generated by the attribute values and therefore the distance to the correct partition will be different from 0. Then, the attribute selected is the one whose distance to the correct partition is smallest. This measure happens to solve the problem of bias of Quinlan's gain ratio measure.

2.8.1.6 Other Splitting Rules

Several other splitting rules have been proposed. Gleser and Cohen (1972) and later Talmon (1986) proposed a measure that relies on dependence, using a combination of mutual information and χ^2 measures. Rounds (1980) has suggested using Kolmogorov-Smirnov distance and test as the splitting criteria. Li and Dubes (1986) proposed a permutation statistic as a splitting criterion. The permutation statistic measures the degree of similarity between two binary vectors: the vector of threshold attribute values and the vector of class labels. Another attribute splitting measure which uses information theory is called binarisation (Bratko and Konenenko, 1986). Binirisation is used to normalise the informativity of attributes with respect to the number of values. This method works by grouping the various attribute values together so that all the attributes become binary. Loh and Vanichsetakul (1988) used a measure that employs statistical hypothesis tests to select a variable for splitting each node and then use linear discriminant analysis to generate linear combination splits. The same idea is followed up by Loh and Shih (1997). Other splitting measures used are based on the minimum description length (Quinlan, 1989; Venables and Ripley, 1994). However, it is not clear how it operates in a statistical domain. Mingers (1989b) has used the G statistic to select among attributes. This is an information theory based statistic that has also been used in contingency tables, and is closely related to Quinlan's information gain measure. Taylor and Silverman (1993) suggested a splitting criterion called mean posterior improvement (MPI), which emphasises exclusivity between offspring subsets rather than equal sized offspring and purity of both daughter nodes (as with *GINI* index). Clark and Pregibon (1991) use the likelihood function to maximize the reduction in deviance produced by each partition. They view the tree as a probability model for the training sample. Van der Merckt (1993) suggested a measure that combines both geometric distance and information gain and argued that such measures are more appropriate for numeric attribute space. A weighted sum approach is considered by Shih (1999), who shows how several splitting criteria can be written as weighted sums of two values of divergence measures. One of them contains the χ^2 and entropy criteria while the other contains the MPI criterion.

2.8.2 Stopping Rules

Stopping rules were originally called pre-pruning and are used to prevent problems like overfitting. These rules are used to stop growing the tree when there is not enough data to make realistic decisions, or when the instances or examples are acceptably homogenous. Esposito *et al.* (1993; 1995; 1997) present these rules as:

1. All observations reaching a node belong to the same class.
2. All observations reaching a node have the same feature vector (but not necessarily belong to the same class).
3. The number of observations in a node is less than a certain threshold.
4. There is no rejection for statistical tests on the independence between feature X_j and the class attribute C (Quinlan, 1986; White, 1997)

Rules 1 and 2 are universally accepted if the decision process adopted for the tree is based on the majority class criterion, i.e. each respective leaf node is labelled by the majority class. Stopping rules are easier and more intuitive than pruning but are not easy to get right (Breiman, *et al.* 1984; Quinlan, 1993). For example, thresholds that are too high can terminate the splitting before its benefit becomes evident, while too low thresholds result in large trees and overfitting resulting in poor generalisation error. Even with the above rules, it remains very difficult to know when to stop growing the tree. One way of getting around these problems is by pruning.

2.8.3 Pruning Rules

Post-pruning is used to prune some branches of the tree after a fully expanded tree is already grown. Most pruning algorithms prune by replacing a subtree by a single leaf node when the estimated error of the leaf replacing the subtree is lower than that of the subtree.

Breiman *et al.* (1984) suggest that instead of using stopping rules, you continue the splitting until all the terminal nodes are pure or nearly pure, thus resulting in a

large tree. Then selectively prune this large tree by getting a decreasing sequence of subtrees. Finally, use cross validation to pick out the subtree which has the lowest estimated misclassification rate.

Pruning is one of the most important part of tree building. DTs that are grown by a recursive partitioning procedure (a greedy look-ahead heuristic) tend to be too large and complex, and are likely to suffer from noise caused by over-fitting (as is the case with all stepwise procedures). The question would be: how much of that model to retain?

The following principal approaches have been used not only to overcome the problem of noise but finding a good structure tree that is likely to give better classification performance on unseen data. All the methods use the bottom-up strategy that starts with the lowest internal node in the tree and prunes those that meet the algorithm's criteria.

2.8.3.1 Minimal Cost Complexity Pruning

This is a cross-validation procedure to trim back the tree. This method is essentially a (multipass) pruning algorithm that was developed by Breiman *et al.* (1984) as part of the Classification and Regression Trees (CART) system. Cost complexity is defined as:

$$R_{\alpha}(T) = R(T) + \alpha\, \text{size}(T) \tag{2.26}$$

where $R(T)$ is a performance measure of a tree, which could be the misclassification error or re-substitution error (cost) of the subtree T, size(T) is the number of terminal nodes of T, and α is the cost-complexity parameter that weighs the relative importance of the tree size to the error rate. For any specified value of α, the "best" subtree produced is the one that minimises the cost complexity measure over all subtrees of T. The method works as follows. First, a sequence of trees (sub-trees) is grown from each node of the induced (grown) tree and the subtrees yielding minimal cost complexity are pruned, successively. The strategy used is bottom-up, i.e., the sequence of trees are all nested and all match at the root node. Then the subtree (the "best" tree) is selected as the final model from the sequence of trees based on its size

and cost complexity measure used. CART either uses two variants of cross validation (CV-0SE, CV-1SE) or two variants of an independent test sample estimate of error (0-SE, 1-SE) to choose the most predictive tree in this sequence. The *k*-SE rule is defined as follows:

Let $\hat{R}(T)$ be the estimated misclassification cost for T and $SE(\hat{R}(T))$ be its estimated standard error. Suppose the tree T_{k0} minimises $\hat{R}(T_k)$. Then the *k-SE* tree T_{k1} is the smallest subtree such that

$$\hat{R}(T_{k1}) \leq \hat{R}(T_{k0}) + kSE(\hat{R}(T_{k0})) \tag{2.27}$$

2.8.3.2 Pessimistic Pruning

Quinlan (1986) developed this top-down tree pruning method in the context of ID3. Like the minimal cost complexity pruning measure, the method does not use a separate pruning set. The estimates that are produced from the training data and based on the resubstitution error are overly optimistic since they tend to be lower than estimates produced by the use of test sets. Pessimistic pruning compares the number of errors introduced by eliminating the branch to the estimated error rate in the subtree using the continuity correction for the binomial distribution plus the standard error of the estimate. One half is added to the number of errors associated with each node in order to get a more accurate error estimate. Quinlan attempts to compensate for the optimistic estimates based on the re-substitution error by adjusting the number of standard errors added to this calculation. Subtrees are not pruned unless their respective corrected number of misclassifications are lower than that of the node by at least one standard error, respectively. One of the advantages of this approach is that the same training set is used for both growing and pruning a tree. The method is also very quick since it only has to make one pass and looks at each node only once. However, the statistical justification for the use of the continuity correction in the estimation of the error rates is not clear.

2.8.3.3 Minimum Error Pruning

Niblett and Bratko (1986) proposed a bottom-up approach seeking for a single tree that minimizes the expected error rate on an independent dataset. Minimum error pruning was first introduced using Laplace probability estimates, and later modified to what was referred to as m-probability estimation (Cestnik and Bratko1991). The parameter m is varied in an attempt to match the degree of tree pruning to properties of the learning domain such as noise. To prune a tree at a node, the expected error rates of its children are first determined, and this is called static error. Dynamic error, defined as the weighted sum of the static error of its children, is then calculated and if the static error is greater than dynamic error the node is replaced by the leaf. One of the disadvantages of this method is the dependence of the expected error rate on the number of classes.

2.8.3.4 Reduced Error Pruning

Quinlan (1983; 1987; 1993) also developed this method as part of the ID3 system. The training data are divided into a *training set* and a *validation set.* For each node in a tree you consider the effect of removing a subtree rooted at that node, making it a leaf, and assigning it majority classification for examples at that node. The performance of each pruned tree is noted over its validation set. The node that most improves accuracy over the validation set is removed. This is continued until further pruning is harmful. This method uses a growing set for induction and the pruning set to estimate the error rate, which is then used to generate and evaluate pruned trees. This is another bottom-up strategy that examines every internal node in the tree and prunes all nodes that it finds when pruned nodes do not decrease the pruning set accuracy of the tree. Reduced error pruning is guaranteed to deliver a small and accurate tree with respect to the pruning set. Unlike MCCP, REP does not build a sequence of trees and thus it has been claimed to be faster. Mingers (1989a) found this pruning technique to be prone to pruning branches that correspond to rare cases. Another drawback of this approach is that one cannot use all the data to build the tree.

2.8.3.5 Error - Based Pruning

Like reduced error pruning, error-based pruning derives error estimates from the training data. This type of method removes the terminal splits which have the same class for each leaf and allows the use of cross validation to estimate the error rate. Breiman *et al.* (1984) suggest the *1*-SE rule by taking the smallest pruned tree whose error rate is within one standard deviation of the minimum. Quinlan (1986) follows a similar approach by comparing the error rate of the tree, say, T_e with the error rate at node t (adding a "continuity correction" of 0.5 to each error count observed at each leaf). He proposes to prune when:

$$\text{Error rate for t} \leq \text{error rate for } T_e + \text{SE(error rate for } T_e)$$

Error-based pruning was also developed as part of the C4.5 system (Quinlan, 1993). Like PP, it assumes that the error in the set of patterns covered by a leaf of a tree follows a binomial distribution and computes a confidence interval on the error counts based on the fact that the binomial distribution is closely approximated by the normal distribution in the large sample case. The upper limit of confidence of the probability of misclassification can then be calculated from an assumed confidence level (default is 25%). The upper limit of this confidence interval is used to estimate the leaf's error rate on fresh data. The predicted error rate comes from multiplying the upper limit confidence by the number of patterns covered by a leaf. If the number of predicted errors is less than that for the subtree containing the leaf, then the subtree is replaced with a leaf. The method further adds a pruning operation called *grafting* by Esposito *et al.* (1997), which allows the node to be replaced by one of its subtrees if this does not increase the estimated error. EPB presents the advantage, with respect to the other pruning techniques, of allowing a subtree to be replaced by one of its branches, However, the assumption that errors in the sample have a binomial distribution is questionable.

2.8.3.6 Other Pruning Procedures

Several other pruning methods have been introduced but their merits have not been examined. Mingers (1986) proposed a bottom-up critical value pruning (CVP)

method similar to REP, which performs the decision about pruning using the " information-gain" that was achieved when growing the tree. It then specifies a critical value which defines the level at which pruning takes place. The full tree is pruned for increasing critical values giving a sequence of trees, and then the best tree is selected on the basis of predictive ability. Crawford (1989) introduced a ***.632*** bootstrap method as an alternative to Breiman *et al.'s* cross validation procedure, which he pointed out has a large variance especially for small training samples. Gelfand *et al.* (1991) claimed that the cross validation method was inefficient and possibly ineffective in finding the optimally pruned tree and modified CART's pruning procedure by proposing a new iterative tree growing and pruning algorithm that is guaranteed to converge. The algorithm divides the training sample into two subsets. The binary tree is iteratively grown on one half of the data and pruned on the other half (using the error rate on the other half), and alternating the roles of the halves in each iteration. This is done until the tree size is unchanged. Their pruning algorithm is simple and (one pass) bottom-up approach which starts from terminal nodes and proceeds up the tree, pruning away branches. Bratko (1994) proposed a pruning method by using dynamic programming, which prune trees optimally and efficiently in the absence of noise. Friedman *et al.* (1996) introduced a pruning technique that is based on analysing the DT as a description of the training instances. Other pruning with costs methods have been proposed (Breiman *et al.*, 1984). Quinlan and Rivest (1989) and later Wallace and Patrick (1993) proposed tree construction and pruning methods based on the minimum description length (MDL). Recently, Oates and Jenson (1998) proposed a method based on the idea of the permutation test called randomised pruning. For each subtree, the probability $\hat{p}$ that an equally or more accurate subtree would have been generated by chance alone is computed. If $\hat{p}$ is greater than a certain threshold (the authors suggest 0.05) the subtree is discarded and replaced by a leaf node.

Approaches other than pruning for improving accuracy of a tree have been proposed and carried out by several authors. Buntine (1992) discussed using smoothing and averaging techniques as one alternative to pruning. Oliver and Hand (1993) and Shannon (1998) and Shannon and Banks (1998) use a set of DTs and then average

them as an alternative method to pruning. Murthy *et al.* (1993) use a technique called tree balancing. A balanced tree is where no leaf is much farther away from the root than any other leaf. Different balancing schemes allow different definitions of "much farther" and different amounts of work to keep them balanced). Tree balancing can be applied after pruning to reduce the depth of the tree without affecting the accuracy of the tree. Brodley and Utgoff (1995) use multiple attributes per test as an alternative to basic pruning, i.e., using splits that contain more than one attribute at each internal node

Empirical comparisons of pruning methods have been carried out by Monago and Kodratoff (1987) who found that pruning methods that use a separate pruning set performed better than others that did not. However, his claim was questioned by Esposito *et al.* (1993) on both methodological and empirical grounds, and found not to be true. In particular, Monago and Kodratoff (1987) used analysis of variance (ANOVA) to test for statistically significant differences between pruning methods instead of paired two-tailed t–test (which had been used by previous investigators). Nonetheless, as with splitting rules, there is no single best pruning algorithm given the overfitting avoidance and bias issues that were raised by Schaffer (1993).

Just as in the case of splitting criteria, no single pruning method has been adjudged to be superior to the others. The choice of a pruning method depends on the size of the training set, availability of extra data for pruning, and so on.

2.8.4. Classification and Error Rates

A DT is used rather like a pinball machine. Given a new instance, an unknown object is classified by starting at the top of the tree. The various tests deflect them one way or another. This will put you in one of the terminal nodes (leaves) of the tree. The predicted class is just that of the majority of the observations that were used to train the tree originally. A tree misclassifies an object if the classification output by the tree is not the same as the object's correct class label.

DT performance is usually given in terms of some variant of misclassification error rate.

Let *f(x)* be the distribution of measurement vectors x and $f(c \mid x)$ the probability of belonging to class c at x. A classification rule estimates $f(c \mid x)$ and forms a defined region R_c in which the rule classifies points to class c. Thus, the error rate for such a classifier is given by $\sum_c \int_{R_c} [1 - f(c \mid x)]f(x)dx$.

Alternatively, suppose you want to predict a response variable of interest Y, given a set of attribute variables $\boldsymbol{X}$, to produce an output $\hat{Y}$. Then, the loss or cost incurred due to the prediction could be defined as (Chou, 1991):

$$L(Y, \hat{Y}) = \begin{cases} 1 & \text{if } Y \neq \hat{Y} \\ 0 & \text{if } Y = \hat{Y} \end{cases} \qquad \text{(misclassification error rate), or}$$

$$L(Y, \hat{Y}) = \left\| Y - \hat{Y} \right\|^2 \qquad \text{(squared error)}$$

These are the same for binary Y.

Due to the difficulty of computing the misclassification rate, it is usually estimated from either the training sample or the test sample. The misclassification error rate is estimated by the ratio of samples misclassified to the total number of samples used to do the test.

There are many types of classification error rates. These are defined below:

Sample error: Given N examples and the number of times that the tree misclassifies the examples, say, E. The error rate for the tree is then E/N. The sample error is obtained from an independent test sample.

Instead of summing terms that are either zero or one as in the error-count estimator, the smoothed error rate uses a continuum of values between zero and one in the terms that are summed. The resulting estimator has a smaller variance than the error-count estimate. Also, the smoothed error rate can be very helpful when there is a tie between two competing classes.

The overall error rate is estimated through a weighted average of the individual group-specific error-rate estimates, where the prior probabilities are used as the weights.

When the input data set is an ordinary data set and no independent test sets are available, the same data set can be used both to define and to evaluate the classification criterion. The resulting error-count estimate has an optimistic bias and is called an resubstitution error rate.

Generalisation error rate: Given a probability measure P_{XY} and a loss function $l : y \times y \rightarrow \Re^{+}$, the generalisation error $G[h]$ of a function $h : x \rightarrow y$ is defined by:

$$G[h] = E_{XY}[l(h(X), Y)] \tag{2.28}$$

Cross validation error: This type of error is used to estimate the error for a tree growing method and it attempts to remove the error rate bias. When using cross validation (CV), you divide the training set into N partitions. Do N experiments: each partition is used once as the validation set, and the other N-1 partitions are used as the training set. Such a technique is very useful when training data is limited.

Average conditional error rate: The true error rate of a rule can be written as $\int e(x)f(x)dx$, where $e(x)$ is the conditional probability of error at x given the correct data set, and $f(x)$ is the overall mixture distribution (Hand, 1997).

Other performance criteria like the computational speed (the time spent learning the tree) and storage (space), depth of the tree (number of splits) and the number of terminal nodes have been used for tree-based models.

2.8.5 Decision Tree Algorithms

The two most standard and leading DT algorithms are the CART system, that of Breiman *et al.* (1984) and ID3 (Quinlan, 1986; 1987) and its successors C4.5 and C5.0 by Quinlan (1993; 2002), respectively. They are both *TDIDT* (Top Down

Induction of Decision Trees; Section 2.2) approaches which induce axis-parallel binary DTs; they use feature vectors for the input but a tree structure for the decision rules that they build up. Post-pruning algorithms are used for both systems.

Many variants of DT algorithms have concentrated on DTs in which each node checks the value of a single attribute. This class of DTs may be called axis–parallel because the tests at each node are equivalent to hyperplanes that are parallel to axes in the attribute space, i.e., they correspond to partitioning the parameter space with a set of hyperplanes that are parallel to all the features axes except for the one being tested and are orthogonal to that one. An example of such a DT has already been given in Figures 2.2 (a) and 2.2 (b), which shows both the tree and the partitioning it creates in a 2-D attribute space. In axis–parallel decision methods, a tree is constructed in which at each node a single parameter is compared to some constant. If the feature value is greater than the threshold, the right branch of the tree is taken; if the value is smaller, the left branch is followed.

There are DTs that test a linear combination of the attributes at each internal node. They allow the hyperplanes at each node of the tree to have any orientation in parameter space. (Murthy and Salzberg, 1992; Murthy *et al.*, 1993). Mathematically, this means that at each node a linear combination of some or all the parameters is computed (using a set of feature weights specific to that node) and the sum is compared with a constant. The subsequent branching until a leaf node is reached is just that used for axis parallel trees. Since these tests are equivalent to an oblique orientation to the axes, we call this class of DTs *oblique* DTs. Note that oblique DTs produce polygonal (polyhedral) partitioning of the attribute space. For obtaining oblique decision, DT program OC1 (Murthy *et al.*, 1993; Murthy *et al.*, 1994) is such an algorithm. OC1 uses a hill-climbing search procedure (i.e. maximising the goodness of split) and chooses multiple attributes for testing trees.

Oblique DTs are considerably more difficult to construct than axis parallel trees because there are so many more possible planes to consider at each node. As a result, the training process is slower. However, they have a major advantage over other methods: they often produce very simple structures that use only a few

parameters to classify objects. It is straightforward through examination of an oblique tree to determine which parameters were most important in helping to classify objects and which were not used.

2.8.5.1 ID3

The ID3 algorithm (Quinlan, 1979; 1983; 1986) equips a new feature to the basic DT algorithm. This is called *windowing*. Windowing can be used if the training set is very large. A subset of the training set called the window is chosen randomly to build an initial tree. The remaining input cases are then classified using the tree. If the tree gives correct classification for these input cases then it is accepted for the entire training set and the process ends.

If this is not the case then a selection of incorrectly classified instances are appended to the window and the process continues until the tree gives correct classification for the whole set. Empirical evidence suggested that a correct DT was obtained more quickly by the windowing method than by creating a tree from the entire training set. However it is suggested by Vadera and Nechab (1994) that the advantages of windowing are negligible and Niblett (1987) has indicated that the windowing method does not always guarantee to find a correct DT unless the window uses the entire training set.

The basic ID3 algorithm, including the window component, is given below:

1. Select at random a subset of the training set (the "window set") of any size. Windowing generates the initial tree from a subset of the data and uses the rest of the data to modify the tree as necessary to account for any statistical differences between the data and the initial subset.
2. Apply a computer learning system algorithm to generate a rule (DT) for the current window set.
3. Test the tree over the rest of the training set.
4. Scan the whole database, not just the window, to find if there are exceptions or misclassified examples to the latest rule.

5. If there are exceptions, insert some of them into the window (possibly replacing existing examples).
6. Stop in branch if pure node is attained. Repeat steps 2 –5; otherwise stop and display the rule.

All the probabilities used to estimate the informativity of the attributes are approximated with relative frequencies taken from the training set.

ID3 has the strength of being fast yet suffers from the fact that it uses non-probabilistic rules. ID3 uses a "greedy" search approach, which performs no backtracking, i.e., once an attribute is selected as an internal node, the algorithm does not go back to change this decision. Therefore, ID3 is susceptible to the usual risk of converging to locally optimal solutions that are not globally optimal. Another limitation of ID3 is that the DT produced overfits the training examples (also known as the problem of small splits) because it performs a stepwise splitting that attempts to optimize at each individual split, rather than on an overall basis (McKee, 1995). This leads to DTs that are too specific. From statistical point of view, very small (or overfitted) groups are quite likely to be chance occurrences and therefore unreliable for predicting new sets of data. Furthermore, ID3 has the weaknesses of being unable to deal with contradictory examples. Finally, ID3 is over-sensitive to small alterations to the training set.

2.8.5.2 C4.5

The system is a successor to ID3 and was developed by Quinlan (1993). It provides two choices of classifier forms; DTs or rulesets. It also performs the window selection method as in ID3 (but with some refinements), and then "greedily" searches over all the attributes to produce a sequence of optimal univariate splits (on both numerically ordered or ordinal variables and on nominal variables), in a top-down recursive divide-and-conquer strategy, so that a large tree is grown. Numeric attributes are handled using the attribute sub-setting method. The tree starts as a single node with all the training examples and uses recursive partitioning. C4.5 uses information gain or information gain ratio measures as splitting rules. The split that

yields the highest information gain or gain ratio is selected. C4.5 follows two types of pruning. During tree construction C4.5 requires that a split results in at least two branches having a minimum number of instances. The default value for the minimum weight is 2, but can be changed by a program option. Larger values of minimum weight help prevent overfitting of noise. The main pruning procedure C4.5 follows is error-based and it is based on statistical heuristics (no cross validation is used) and relies on the estimated confidence intervals on the training set, as already described in Section 2.2.3.

Large DTs can sometimes be very difficult to understand. An important feature of C4.5 is its mechanism to convert DTs into collections of rules called *rulesets*. The hypothesis space of C4.5 is within the disjunctive normal form (DNF) formalism, i.e., for each DNF, the conditions along the branch represent conjuncts and the individual branches can be seen as disjuncts. Each branch forms a rule with a conditional part and a conclusion. The conditional part is a conjunction of conditions. In other words, you write a rule for each path in the DT from the root to a leaf. In that rule the left-hand side (LHS) is easily built from the labels of the nodes and the labels of the arcs. The resulting rules set can then simplified: Let LHS be the left-hand side of a rule and let LHS* be obtained from the LHS by eliminating some of its conditions. We can replace LHS by LHS* in this rule if the subsets of the training set that satisfy respectively LHS and LHS* are equal. The DT example (See Figure 2.2 (a)) can be converted to the following rules:

IF (Outlook = Sunny) and (Humidity >75)
THEN Don't Play Golf

IF (Outlook = Sunny) and (Humidity <=75)
THEN Play Golf

and so on.

For a detailed discussion of rule learning, the reader is referred to Section 2.7.

C4.5 provides a choice of classifier forms - DTs or rulesets. In many applications, rulesets are preferred because they are simpler and easier to understand than DTs, but the methods for finding rules used in C4.5 are slow and memory-hungry.

A major weakness of C4.5 is that the information gain measure selection criterion it uses has a tendency to favour many valued attributes. However, Quinlan offers a solution to this by introducing the information gain ratio measure. As mentioned in Quinlan (1986), this ratio may not always be defined; the denominator may be zero or it may tend to favour attributes which the denominator is very small. Also, the gain ratio measure also discriminates the selection of attributes with many uniformly distributed values. However, the experiments described in (Quinlan, 1988) show improvement in tree simplicity and prediction accuracy when gain ratio criterion is used.

C4.5 always subdivides the data subsets until no single exception remains. The resulting subsets may become so small that partitioning them further would have no statistically significant basis. By creating a branch for each attribute value, C4.5 encounters the problem of over-branching caused by unnecessary partitioning of the training data.

2.8.5.3 See5/C5.0

See5/C5.0 represents a complete rethink of Quinlan's C4.5 algorithm for generating DTs and rulesets, and the improvement is dramatic. Also, C5.0 is for classification only; there is no regression tree. It uses multiple splits rather than binary splitting. The Unix system C5.0 and its Windows counterpart See5 are superior to Quinlan's earlier system C4.5 in several important ways.

C5.0 is more than two hundred and forty times faster than C4.5 on the coding data. It uses less memory (5.6Mb versus more than 6.0Mb for C4.5), and produces a more accurate ruleset into the bargain (Quinlan, 2002).

C5.0 incorporates several new facilities such as variable misclassification costs. In C4.5, all errors are treated as equal, but in practical applications some classification errors are more serious than others. C5.0 allows a separate cost to be defined for each predicted/actual class pair; if this option is used, C5.0 then constructs classifiers to minimise expected misclassification costs rather than error rates.

C5.0 has several new data types in addition to those available in C4.5, including dates, times, timestamps, ordered discrete attributes, and case labels. In addition to missing values, C5.0 allows values to be noted as not applicable. Further, C5.0 provides facilities for defining new attributes as functions of other attributes.

C5.0 is also easier to use. Options have been simplified and extended to support sampling and cross-validation, for instance.

2.8.5.4 CART

CART is an acronym for Classification and Regression Trees, a decision-tree procedure introduced by Breiman *et al.* (1984) based on Friedman's foundation (Friedman, 1977). It uses a "greedy" approach and several single-variable (univariate) splitting criteria on both numerically ordered or ordinal variables and on nominal variables - Gini, twoing, ordered twoing and least squares and least absolute deviation for regression trees - and one linear combination split on numerically ordered variables method. Linear combination splits are in the form $\sum_i a_i X_i \leq c$ versus $\sum_i a_i X_i > c$, where c is the "cut-off" point and c, $\{a_i\}$ are constants that are chosen to maximise separation of the sample. The default GINI method typically performs best but, given specific circumstances, other methods can generate more accurate models.

In CART, pruning is done via the test sample or cross validation, utilising the cost complexity pruning technique. The idea is to estimate the misclassification cost for each subtree with the test sample or cross validation and select the subtree with the smallest estimated cost.

The CART system is quite efficient. It generates results much faster than other tree algorithms such as C4.5 and non-tree classification methods, such as neural nets. C4.5 has since been superseded by C5.0 which is much faster than CART. However, C5.0 still does not have the cross-validation capability to choose the best tree.

CART allows only either a single feature or a linear combination of features at each internal node. Linear combination splits and nominal variables with many

categories are computationally very expensive, as it requires generation of multiple auxiliary trees. There is also no guarantee of global optimality with linear combination splits. However, they are more nearly optimal than if linear combination is not used. Also, it should be noted that considering the use of linear combination splits increases the risk of overfitting substantially. One other important criterion for classifiers is computational speed which is affected by many factors. For example, with most tree-based algorithms, a 10 fold cross validation increases running time by a factor of 10.

2.8.5.5 Other Systems

Several other tree-based systems have been developed for supervised learning. Most of them re-implement some parts of the "classical" CART and C4.5 algorithms. We do not describe all of them here in detail but some of them deserve mention.

CAL5

This system is due to Müller and Wysotzki (1994; 1996) and was designed specifically for continuous and ordered attributes. It was also designed to convert real-valued attributes into discrete ordered attributes (intervals) using statistical splitting methods. The intervals are automatically constructed and adapted to establish an optimal discrimination of the classes in the feature space. However, CAL5 does have a procedure to handle unordered discrete valued attributes. The trees are constructed top-down by stepwise branching with new attributes to improve the discrimination of classes.

A quotient is used as the evaluation function for splitting given by the following measure:

$$\text{quotient}(N) = \frac{A^2}{A^2 + D^2} \tag{2.29}$$

where N is the internal node in the tree construction process; A^2 is the mean value of the square of distances between the centroids of the classes; D^2 is the mean value of the squared variance of the classes with respect to their centroid vector. This is done for continuous attributes only, and an attribute with the least value of the

quotient is selected as the best one for partitioning. The evaluation function requires discretisation at N for each attribute before choosing the best attribute for splitting at that node. Discretisation is done by forming, recursively, intervals and discrete points on the axis for ordered and unordered discrete variables, respectively. This is done until a class decision can be made at a given level of confidence. Trees grown by CAL5 can also use Quinlan's gain measure when choosing the best attribute for splitting. The trees are automatically pruned with the help of a threshold and significance level for the estimated class probabilities in an interval. By means of this threshold the user can control the complexity of the tree, i.e., the degree of approximation of class regions in feature space. In addition, pruning occurs during tree construction.

One main strength of this system is its ability to construct either binary discrete valued attributes or attributes with an arbitrary number of discrete values (intervals) and work entirely with either. It has no restriction to binary ones as, for example, in CART or in modifications of ID3. An obvious limitation of this method is the procedure of pruning during learning. It is always difficult to know when to stop growing the tree. It is also difficult to choose an appropriate threshold that determines whether to split a node or not.

QUEST

Quick, Unbiased and Efficient Statistical Tree (QUEST) is a statistical DT algorithm for classification and data mining developed by Loh and Shih (1997). The objective of QUEST is similar to the CART algorithm. The algorithm constructs a tree by linear statistical methods. QUEST uses a variable selection method that is unbiased in selecting variables that have more splits and also much faster than the exhaustive search method of Breiman *et al.*, especially when there are nominal variables with many categories, or when linear combinations splits are employed. The F and χ^2 statistical tests are utilised for this task of variable selection. At each node, an analysis of variance F-statistic is calculated for each ordered variable. The variable with the largest F-statistic is selected and linear discriminant analysis is applied to it to find the best splitting point. Unlike CHAID (described later) and CART, which

handle variable selection and split point selection simultaneously during the tree growing process, QUEST deals with them separately. Pruning is done via test sample or cross validation while missing values are handled by imputation. The strength of QUEST lies in its fast tree construction speed. QUEST has been demonstrated to be much better than exhaustive search methods in terms of variable selection bias and computational cost. In terms of classification accuracy, variability of split points and tree size, however, there is no tree algorithm that has been found to be superior to the others when univariate splits are used. The most obvious limitation at present is its inability to detect pairwise interactions between predictor variables at each node (Loh, 2001).

AID, THAID and **MAID**

Morgan and Sonquist's early 1960's work on AID (Automatic Interaction Detector) created interest in recursive partitioning. AID uses binary splits, chooses the predictor and split that maximizes the variance explained by the binary split, and stops splitting when there is no split that could explain some user-specified minimum fraction of the original full-sample sum of squared deviations. After AID, the same group developed THAID (Theta AID) for a categorical dependent variable (Morgan and Sanquist, 1973), and MAID (Multivariate AID) for a continuous dependent variable (Gillo, 1972). The statistical community's interest in AID was soon tempered however by the realization that the arbitrary stopping rule was not effective in controlling Type I and Type II errors, and AID acquired the reputation of 'a method that could find a significance in any data set'. This started two independent threads of work to put AID on a statistically sound foundation so that it would make only 'real' splits. This led to the birth of CHAID, which is discussed below.

CHAID

One interesting rule-generator is the system described by Kass (1980) called Chi-Squared Automatic Interaction Detection (CHAID). The system partitions the data into mutually exclusive and jointly exhaustive subsets that best describe the class variable. It tends to favour non-binary (multi-way) splits that can paint visually

appealing trees, but that can bog models down with less accurate splits. The best partition is chosen on the basis of a Bonferroni-corrected p-value of the χ^2 statistics for each merged predictor. CHAID's multi-way splitting criterion is based on measuring the association between two variables in a contingency table, and then utilising Pearson's χ^2 statistic to select among attributes. An improved method of calculating the significance is proposed by Biggs *et al.* (1991).

CHAID has the weakness of yielding trees with excessive branching at each node. This can affect the gain in impurity and significantly biases the selection towards nominal variables with many categories (Loh and Shih, 1997). To address this problem, White and Liu (1994) suggested using only binary splits, i.e., splitting a node into branches so as to avoid having to decide what an appropriate number of branches would be. Also, multiway splitting does not make as effective use of the conditional information potentially present in the tree as does binary splitting (Friedman, 1977). However, Kass (1980) seem to think that binary splits are often misleading and inefficient by arguing that they poorly communicate structure in the data if the data more naturally split into more branches. For example, if salaries are vastly different in Swaziland, South Africa and United Kingdom, then the algorithm, ought to separate the three countries all at once when predicting salaries.

2.8.5.6 Further Systems

Several statistical packages have since incorporated tree functions, like S-Plus Tree (Clark and Pregibon, 1992; Venables and Ripley, 1994) and RPART (Therneau and Atkinson, 1997), which implement binary trees for classification and regression problems.

Other types of DT systems have been proposed. The standard algorithm of building DTs uses at each node a test based on one attribute. These type of trees are called univariate trees. Multivariate trees use tests based on linear combinations of attributes, i.e., splits that are based on more than one attribute at each internal node (Brodley and Utgoff, 1995).

Another approach is known variously as dynamic path generation (White, 1977) or a lazy DT (Friedman, 1996) which involves branching an attribute in a way that is best for each individual case. Unlike dynamic path generation, lazy DTs construct the "best" DT for each test instance. The information gain measure is used to choose the appropriate test but there is no pruning done. Missing values are handled by considering only splits on feature values that are known in the test instance.

Incremental induction (modifying the existing knowledge in response to new data or facts) has been proposed by several authors. Incremental decision trees (IDTs) are trees where statistics about the attribute values are maintained at each node without retaining past examples (Schlimmer and Fisher, 1986; Utgoff, 1991; Kalles and Morris, 1996). The basic idea of IDT is to accept new training instances, and to update the tree in response. This is done in two steps. First, by incorporating a training instance into the tree by passing it down the branches until it reaches a leaf. Secondly, by traversing the tree from root to leaves to ensure a best test at each decision node by which to partition the training instances. The main strength of incremental learning is in its lower cost for serial learning than repeated running of a non-incremental learning algorithm.

Several authors have considered constructing tree classifiers that have linear discriminants (Duda and Hart, 1973) at each node in the DT. These type of trees are known as linear discriminant trees. Loh and Vanichsetakul (1988) have considered the use of linear discriminants at each node of the DT for their called the fast algorithm for classification trees (FACT) system. Their method chooses the variables at each stage according to the data and the type of splits desired. Yun and Fu (1983) use a multivariate stepwise regression approach to optimise the structure of the DT as well as to choose subsets of features to be used in the linear discriminants.

Bayesian trees - Another form of tree induction but from a Bayesian theory point of view, have been considered by several authors. The two basic components of this approach consist of prior specification and search. Bayesian trees put independent and identical Dirichlet priors at each node. The priors induce posterior distributions which then guide the model search algorithm (deterministic or stochastic) towards

fitting the tree (Buntine, 1990; Chipman *et al.*, 1998a; 1998b; Denison *et al.*, 1998).. The class posterior distribution is calculated for each terminal node, which makes the use of DTs within a Bayesian framework computationally expensive (Breiman *et al.*, 1984; Buntine, 1992). Trees with largest posterior probability are chosen as good trees. As well as re-implementing parts of the CART and C4.5 algorithms and offering experimental control suites, IND (Buntine, 1992) also introduces full Bayesian and Minimum Message Length (a Bayesian method of inductive inference) methods and more sophisticated search in growing trees.

Option trees - Trees include option nodes, which replace a single decision with a set of decisions. Buntine (1990) introduced options trees as a generalisation of DTs. For split selection, all promising attributes are selected instead of the "best" attribute as most DTs algorithms do. For each selected attribute a DT is built. Once the option trees have been built, averaging is then done on them (Buntine, 1990; Kohavi and Kuntz, 1997). The main difficulty of option trees is that they require a lot more memory than ordinary trees.

Intermediate decision trees (Holder, 1995) are the subtrees of the full (unpruned) DT generated in a breadth-first traversal of the internal nodes (splits) of the DT. The best-first traversal orders the splits based on information gain. Both the depth-first and best-first traversals indicate higher error until a majority of the splits have been performed. The breadth-first traversal, however, quickly achieves a low error, which gradually ascends to the final error level.

Recent approaches have involved the use of fuzzy theoretical methods (Zadeh, 1965; 1994) to create fuzzy trees (Klir and Folger, 1988; Hayishi et al., 1998; Hirota and Pedrycz, 1996; Holve, 1997; Janikow, 1998). By applying such approximate reasoning techniques, more flexible and robust DTs have been created. The fuzzy tree allows for gradual transitions to exist between attribute values, whilst simultaneously maintaining a degree of transparency in how the decision outcome was reached. Current work is being also being undertaken on the creation of fuzzy DTs using a range of hybrid machine learning techniques. Once again, the statistical validity of this approach is open to questions.

Cestnik *et al.* (1987) makes some improvements on the attribute splitting and pruning strategies used by other tree-based systems. ASSISTANT 86's main idea is based on the binarisation of attributes and the growing of binary trees.

The GID3* (Fayyad and Irani, 1991; Fayyad, 1991) system builds on Quinlan's ID3 learning system. Its rules start off like the ID3 and become more and more specific. It uses the gain ratio measure rather than the entropy as a selection measure, but branches only on a subset of values while grouping the rest in one default branch. A selection criterion is then used to select the attribute that "induces" the best partition on the data. Fayyad and Irani (1991) further define a new family of selection measures, called C-SEP, which they argue are better suited for the purpose of class separation.

2.8.5.7 Strengths and Weaknesses of Decision Trees

Strengths of DTs

1. One property that sets DTs apart from all other methods is their invariance to monotone transformations of the predictor variables. For example, replacing any subset of the predictor variables $\{x_j\}$ by (possible different) arbitrary strictly monotone functions of them $\{x_j \leftarrow m_j(x_j)\}$, gives rise to the same tree model. Thus, there is no issue of having to experiment with different possible transformations $m_j(x_j)$ for each individual predictor x_j to try to find the best ones. This invariance provides immunity to the presence of extreme values ("outliers") in the predictor variable space. In addition, DTs incorporate a pruning scheme that partially addresses the outlier (noise) removal problem.
2. A DT is a form of knowledge that is relatively easy for humans to generate and understand from a conceptual view point. However, this is necessarily not true if most attributes are continuous and the tree is very large.
3. DTs are able to generate understandable rules.

4. DTs are non parametric in nature; since they do not assume any underlying family of probability distributions. This makes them more robust than parametric techniques. DTs are capable of generating arbitrarily complex decision boundaries from a given set of training samples.

5. DTs are relatively fast to construct and classification is very fast too. They work for almost all classification problems and can achieve good performance on many tasks.

6. DTs are particularly convenient for handling mixtures of real-valued and nominal features and interactions among them for which there is no well-defined distance metric, which would be required for a nearest neighbour of kernel estimation approaches. They can also handle a large number of features.

7. It is easy to read or interpret small trees. They also generate understandable rules no matter how complicated the units are; it is generally easy to follow any one path through the tree, so explaining the decisions along the way is easy.

8. The computation cost for each split is inexpensive; programs are relatively fast to run.

9. Once a DT has been constructed, classification of future cases is fast and simple.

10. DTs provide a clear indication of which attributes are most important for prediction or classification.

Weaknesses of DTs

1. The principal limitation of DTs is that in situations not specifically advantageous to them, their performance tends not to be competitive with other methods that might be used in those situations.

2. Another problem with DTs is instability. Changing the values of just a few observations can dramatically change the structure of the tree, and substantially change its predictions. This is especially the case for large trees.

3. It is also hard to use DTs for problems involving time series data unless a lot of effort is put into presenting the data in such a way that the trends are made visible.

4. DTs suit discrete attributes (relatively poor performance when applying them to mainly continuous attributes problems). Some problems with continuously-values attributes or classes may not be easily discretized.

5. DTs are prone to errors in classification problems with many classes and a relatively small number of training examples.

6. A greedy algorithm is non-backtracking. Once a data set is partitioned into multiple subsets, knowledge across the subsets cannot be explored.

7. They have problems with large number of missing data. Most of the available methods for handling missing attribute values in trees are somewhat clumsy.

8. The process of growing a DT can be computationally expensive. Each attribute which is considered a candidate for splitting field must be sorted before its best split can be found. In some algorithms, combinations of fields are used and a search must be made for optimal combining weights. Pruning algorithms can also be expensive since many candidate sub-trees must be formed and compared.

9. DTs do not deal with non-rectangular partitioning of the data space well. Most DT algorithms only examine a single attribute at a time. This leads to rectangular classification boxes that may not correspond well with the actual distribution of records in the decision space.

10. In some instances, especially when the number of classes is large, DTs can cause some nodes to have *overlap* classes, i.e., the number of terminal nodes is much higher than the number of actual classes, thus increasing the search time and space. In many situations this multiplicity reflects variation in posterior probabilities.

Chapter 3

Missing Values

3.1 Overview and Problems Caused by Incomplete Data

Given a large database it is unlikely that all the information will be complete for each case. This seems especially common in medical and social sciences. Rates of less than 1% missing data are generally considered trivial, 1-5% manageable. However, 5-15% require sophisticated methods to handle, and more than 15% may severely impact any kind of interpretation (Pyle, 1999). Incomplete data could be caused by unit nonresponse (where no data could be collected from the sampled unit) or item nonresponse (where partial data is collected for the unit, but some items are missing). In panel studies, where persons are interviewed several times, causes of incomplete data could be the first and subsequent interview or "wave" nonresponse (where a subject is missing for one or more waves of the panel survey) and attrition or dropout (where a subject leaves a panel survey and does not return).

There are many patterns of missingness in data (Cohen and Cohen, 1983). The pattern simply defines which values in the data set are observed and which are missing. These can be classified into general pattern and special patterns (univariate missing data, unit nonresponse and monotone missing data). The general or arbitrary pattern is when any set of variables may be missing for any unit. Univariate missing data occur when missing values are confined to a single variable, i.e., only one variable is subject to nonresponse. Another pattern could be where a block of variables is missing for the same set of instances, and the remaining variables are complete ("unit nonresponse"). The third special pattern occurs if a variable, say Y_j, has missing values missing then the other variables,

say $Y_{j+1},...,Y_p$, have missing values as well (monotone pattern). The pattern of monotone missing data arises commonly in longitudinal data subject to attrition.

Incomplete data can cause two major problems in general: decrease in statistical power of hypothesis tests and bias in results, especially parameter estimates because the sample size for the incomplete data is less than it would be if the data were complete (no missing values). The concept of power in statistical theory is defined as the probability of rejecting the null hypothesis given that the null hypothesis is false. In this context, statistical power is the ability of a statistical test to discover a relationship in a given set of data. Power depends on the type of test, increases with increasing sample size, effect size, and significance level, and declines with increasing sampling variance. Kim and Curry (1977) showed experimentally how statistical power requirements could be associated with increases in proportions of missing data.

Second, missing data can bias results, especially parameter estimates. All these effects of missing data depend upon why the data are missing and the method used for handling missing data in analyses. Missing data are also problematic because most statistical packages require a value for each variable. When a data set is incomplete, the data analyst has to decide how to deal with it or the algorithm being used must have a mechanism for dealing with it.

Another problem with missing data is that they make common statistical methods inappropriate or difficult to apply (Rubin, 1987). For example, when missing data are present in a factorial analysis of variance the design is unbalanced (i.e. unequal number of instances in cells of a design). Consequently, the standard statistical analysis that is appropriate for balanced designs is no longer appropriate under this condition. Even if data are assumed to be missing in a completely random fashion, the proper analysis is complicated because the effects are no longer orthogonal.

Finally, valuable resources are wasted as a result of missing data. Time and funding spent on subjects who subsequently leave a study and/or produce missing data represents a loss. Such loss is a particular concern in longitudinal research, large

scale assessments and high-stake studies, and surveys that ask sensitive information or target respondents who are not accustomed to responding to opinions surveys.

Handling data with only partial information on some variables is a serious problem for many large-scale surveys or panel studies, particularly when statistical techniques whose methods are built on the assumption that data are complete were intended to be used. Missing values may distort the results, imbalance the study design and produce biased estimates. Various ways of handling missing values have been extensively studied (Afifi and Elashoff, 1966; Hartley and Hocking, 1971; Beale and Little, 1975; Rubin, 1976; Dempster *et al.*, 1977; Kim and Curry, 1977; Everitt, 1984; Little and Rubin, 1987; Schaffer, 1997). These methods for handling data will depend the amount of missing data; on why the data is missing (the mechanism); what data are missing (pattern of missingness); the covariance between the variables; and what particular function of the population parameters is being estimated (producing sound estimates of the parameters of interest). However, the two most common tasks when dealing with missing values are to investigate the pattern of missingness to get an idea of the process that could have generated the missing data and to produce sound estimates of the parameters of interest, despite the fact that the data are incomplete. The law generating the missing values, i.e., the missing data (values) mechanism, seems to be the most important task since it facilitates how the missing values could be estimated more efficiently. Formally, this law is the conditional distribution of the missing indicators, given all the variables considered.

3.2 Types of Missing Data Mechanisms

Almost all techniques suggested in the literature assume that information is missing randomly (Hartley and Hocking, 1971; Little and Rubin, 1987; Schafer, 1997). But the simple dichotomy - random versus non-random is often not sufficient. The most appropriate way to handle missing or incomplete data will depend on how data points became missing. All the causes for missing data fit into different classes, which are based on the relationship between the missing data mechanism and the

missing and observed values. These classes are important to understand because the problems caused by missing data and the solutions to these problems are different for the individual classes. We begin with the examination of various patterns of data.

It is useful to consider the probability law generating the missing values or the mechanism by which missing data arise. By missing mechanism we mean the frequency distribution of different categories of missing patterns such as missing on one variable, missing on two related variables, and so on. Rubin (1976), and Little and Rubin (1987) distinguish missing data generating processes with respect to the information they provide about the unobserved data. Formally, this missing mechanism is the conditional distribution of the missing indicators, given all the variables considered. The most important issue is whether the missingness is related to the values of other variables. For example, whether information is missing or not on a given variable say, Y:

a) may be unrelated to the values of that variable or to the values of other variables in the data set (independent of all variables);

b) is dependent on the value of another variable, say, X;

c) is dependent on the values of itself, that is, Y;

d) may be determined by values not observed in the given data set;

e) is a product of a particular combination of two or more variables.

Little and Rubin (1987) view response as a random process and further define three types of missing data mechanisms. The processes are that data are missing completely at random (MCAR), missing at random (MAR), or informatively missing (IM), which is also known as non-ignorable or not missing at random.

To discuss the effects of missing data, Little and Rubin (1987) introduced a missing-data indicator matrix $\mathbf{R}$, with (i,j)th element R_{ij} where $R_{ij} = 1$ if X_{ij} is observed and $R_{ij} = 0$ if X_{ij} is missing. The three missing data mechanisms are therefore defined as:

MCAR the distribution of **R** does not depend on the observed or missing values of the variable's data, say, Y. MCAR is the strongest assumption that can be made about the missing data mechanisms. It assumes a pure random missingness. An example of MCAR is when subjects are absent from a measurement session for reasons entirely unrelated to that variable or to the values of other variables in the data set being measured. The key idea is that missingness is unrelated to outcome, i.e. missing instances are no different than non-missing instances, in terms of the analysis being performed. Data that are missing because a researcher dropped test tubes or survey participants accidentally skipped questions are likely to be MCAR. Thus, these instances can be thought of as randomly missing from the data and the only real penalty in failing to account for missing data is loss of power. Simple approaches like mean imputation or regression-based imputation are satisfactory only for data that is MCAR.

MAR the distribution of **R** depends on Y only through the observed values, but is not related to the value that should have been observed for that data point. It is a conditional MCAR or MCAR is a special type of MAR. Accounting for the values which "cause" the missing data will produce unbiased results in an analysis. Thus, it is a weaker assumption. An example of MAR is a school-based study where the probability of completing the questionnaire can be explained by grades. Another example of this type can be found in a situation in which respondents are asked if they voted in the previous elections but some were ineligible to vote because of age. Full maximum likelihood and Bayesian approaches can handle data that is MAR (and also MCAR).

IM the distribution of **R** depends on the unobserved values (and possibly the observed); missingness depends on the value that should have been observed at that data point. This type of mechanism is also known as Non-Ignorable (NI) or not missing at random (NMAR) or missing not at random (MNAR), and it is the most problematic and hardest to deal with. Since the missing data depends on events or items which the researcher

has not measured, this is a damaging situation. An example of such a mechanism is a study of risk behaviour where nonresponse is directly related to behaviour or respondents with excessively higher income may be reluctant to reveal their level of income. Approaches to dealing with IM data are an area of active current research.

The reader is referred to Table 3.1 for more detailed definitions of these concepts.

Table 3.1: Missing data hierarchy (Little and Rubin, 1987; Schafer, 1997)

1. Missing completely at random (**MCAR**):

$$P(R|Y,\varphi) = P(R|\varphi) \text{ for all } Y$$

2. Missing at random (**MAR**): (non-response mechanism is *ignorable*)

$$P(R|Y,\varphi) = P(R|Y_{observed},\varphi) \text{ for all } Y_{missing}$$

3. Informatively missing (**IM**): (non-response mechanism *is non-ignorable*)

$P(R|Y,\varphi)$ depends upon $Y_{missing}$ possibly also $Y_{observed}$

where, R = missing data indicator matrix

Y = data matrix

φ = unknown parameters

A key distinction is whether the mechanism is *ignorable* (i.e. MCAR or MAR) or *non ignorable* (i.e. IM). Ignorability is a desirable property, can be guaranteed in many settings when the experimenter has control over data that is missing. There are excellent techniques for handling ignorable missing data. However, non ignorable missing data are more challenging and require a different approach. The statistics literature has investigated a few techniques for directly handling non-ignorable missingness, but the techniques are problem-specific and sensitive to prior assumptions (Little and Rubin, 1987).

These distinctions are important because data that are MCAR produce unbiased estimates even with rather primitive analysis methods. Data that are MAR will produce unbiased estimates, if a model and estimation technique is used that renders the missingness mechanisms *ignorable*. When data are IM, an analysis

method must be used that includes both a model for the observed data, and a model for the missingness mechanism. For data that are MCAR or MAR, general software is available that produces unbiased estimates using all the available information. For data that are IM, there are usually no easy solutions.

There is no doubt that these concepts play a key role in the theory of missing data adjustments (as evidenced by the number of papers where they are cited). For example, complete-case analysis often (not always) makes an MCAR assumption, and likelihood methods that ignore the missing data mechanism assume MAR. These terms have turned out to be incredibly helpful both theoretically (to classify the type of analyses that are needed) and practically (to show that nearly all *ad hoc* methods implicitly assume MAR). These terms have also shown that there never is any direct evidence against MAR in observed data. Thus, any IM modeling will rely on external (sometimes perfectly reasonable) assumptions. Making assumptions underlying methods explicit seems to be very important, since prior to that people carried out missing data adjustments without any awareness of what was being implicitly assumed about the mechanism.

Data can provide evidence against MCAR (Little and Rubin, 1987). The data cannot generally distinguish between MAR and IM without distributional assumptions, unless the mechanism is well understood (for example, right censoring is IM but is in some sense known). The term "right censored" implies that the event of interest, *i.e.,* the time-to-failure, is to the right of our data point. In other words, if the units kept on operating, the failure would occur at some time after our data point (or to the right on the time scale). It does imply that most analyses rely on un-testable assumptions. There are methods that attempt to set up bounds for parameters that make the fewest possible assumptions (for example, assuming all missing binary outcomes are zero, or all are one) but such methods have the disadvantage of giving equal credibility to extreme or more plausible models, and they can only be applied in restrictive settings. We would say the only way to avoid assumptions in many missing-data problems is not to have any missing data.

Graham and Donaldson (1993) referred to missing data mechanisms as "accessible" and "inaccessible". An accessible mechanism is one where the cause of missingness can be accounted for. The term accessible is related to the term ignorable, except that accessible refers only to the missing data mechanism whereas ignorable refers to the combination of the mechanism and the data analysis. These situations encompass MCAR and most MAR circumstances. An inaccessible mechanism is one where the missing data mechanism cannot be measured, i.e. the missingness is dependent on an unobserved variable, and hence, the mechanism cannot be included in the analysis. These situations include non-ignorable mechanisms and MAR mechanisms where the missingness is known, but is not measured.

Hand (2000), introduces the high level and low level terminology, interchangeably, as another type of missing value mechanism. High level refers to entire records being missing, so that the sample distribution is biased relative to the population. Low level refers to individual fields within the records being missing (Hand *et al.*, 2001). So in a survey, high level missing would mean that some people answer no questions at all. In fact, it is not even known that they were supposed to be in the survey. Low level means some people refuse to answer some questions (for example, they did not give their age, though they did answer other questions).

Knowledge of the missing data mechanism is the main element in determining a treatment for missing data, and largely determines the performance of this treatment. It is however impossible to verify the MCAR assumption and the causes of missingness in practice without additional information. Still one can investigate the missing data patterns in the data and use the available information to make reasonable guesses about the mechanism.

It has been shown that the pattern and mechanism of missing data have greater impact on research results than does the amount of data missing (Little and Rubin, 1987; Graham and Donaldson, 1993; Roth, 1994; Tabachnick and Fidell, 2001). Hence, both are critical issues a researcher must address before choosing an appropriate procedure to deal with missing data. Randomness of data has also been

shown to influence greatly the accuracy of missing data techniques (Roth, 1994; Little and Rubin, 1987; Graham and Donaldson, 1993).

It is useful to distinguish the pattern of missing data and the missing data mechanism. The pattern simply defines which values in the data are observed and which are missing. Missingness confined to a single variable is an example of a univariate pattern. Missing data mechanisms, on the other hand, concerns the reasons why values are missing, and in particular the question whether the fact that variables have missing values is related to the underlying values of the other variables in the data set.

3.3 General Approaches to Dealing with Missing Data

The history of the development of missing data techniques (MDTs) can be divided into three periods (Schafer, 1997). In the first period, prior to 1980, most methods dealing with incomplete data were *ad hoc*. In the second period, principled methods such as the full information maximum likelihood (FIML) and the Expectation-Maximization (EM) algorithm began to appear. The late development of MDTs began in the late 80s and early 90s; it was characterised by the introduction of multiple imputation methods to overcome the drawbacks of single imputation methods. In this section, we will present a brief overview of existing methods for dealing with missing data. These methods are divided into two categories: ignoring and discarding data and imputation (which could either be single or multiple imputation).

3.3.1 Ignoring and Discarding Data

3.3.1.1 Listwise Deletion

Incomplete data are often dealt with by using several general approaches, like, deleting the cases with missing data (listwise or instancewise deletion methods) which aim to modify up the data so that they can be analysed by methods designed

for complete data. Listwise or pairwise deletion methods have been used when it is assumed (or the randomness tests show) that the pattern of missing data does not deviate significantly from the random model (Kim and Curry, 1977; Muthèn *et al.*, 1987; Arbuckle, 1996). Such an approach is *ad hoc* and has little theoretical justification. Hence, the implementation of several new theory-based methods, which tend to be more efficient (under a missing completely at random process) and less biased (under missing at random process).

Basically, listwise deletion (LD) means that any individual with missing data on any variable is deleted from the analysis under consideration. This approach can drastically reduce the sample size since it can sacrifice a large amount of data leading to a severe lack of statistical power (Roth, 1994). It can even lead to complete case loss if many variables are involved. However, due to its simplicity and ease of use, LD is the default in most statistical packages.

3.3.1.2 Pairwise Deletion

Pairwise deletion (PD) is a form of listwise data deletion but is used when calculating any statistic that is based on pairs - such as a correlation. PD is different to LD in the sense that more data is used. So, anyone missing data on a variable involved in a pair is deleted. This means that elements in a correlation matrix may be based on different instances and possibly different sample sizes. Interpreting the correlation matrices can be difficult at times because different samples are used for each statistic, which can result in mathematically inconsistent correlations or a covariance matrix that is not positive definite (Kim and Curry, 1977). Both approaches assume that the missing data are missing completely at random (Little and Rubin, 1987) - a very stringent assumption that is difficult to defend.

For data that are missing completely at random (MCAR), PD and LD estimates are consistent (Little and Schenker, 1995). However, under MCAR, PD was found to be marginally more efficient than LD (Arbuckle, 1996). In other words, the residual mean squared errors (RMSE's) of parameter estimates by PD did not appear to be larger than those obtained under LD. Kaplan (1995) and Muthèn *et al.* (1987) found that under MCAR, the PD approach yielded unbiased estimates. If the data are

missing at random (MAR), PD and LD estimates can also be biased (Brown, 1994; Little and Schenker, 1995). One final shortcoming of PD is that it does not provide standard errors of parameter estimates or tests of model fit (Arbuckle, 1996). One advantage of LD is its simplicity, because standard statistical analysis can be applied without modification for incomplete data. Second, such an approach is non-parametric, i.e., it makes no assumptions about the distribution of data. Thirdly, in a few special cases, LD is the statistically optimal method and finally, LD has been shown to yield correct (although perhaps not efficient) inferences under MCAR.

One of the limitations of LD is that it is inefficient and can also introduce biases if missingness is not MCAR (Little and Rubin, 1987). Also, LD ignores possible systematic differences between complete cases and incomplete cases. The standard errors will generally be larger in the reduced sample because less information is utilized. When using LD you can get biased estimates if the reduced sample is not a random sub-sample of the required sample. Loss of information is another weakness of instance deletion. For example, in discarding incomplete instances; one may discard unacceptably a large portion of instances, especially in many variate problems. Kim and Curry (1977) have shown how randomly deleting 10% of the data from each variable in a matrix of five variables could easily lead to eliminating 41% of instances from the analysis. It is also unclear which set of instances should be used for a particular analysis. However, all the complete instances are used most of the time. The LD approach has been implemented as the default method of handling incomplete data by many statistical procedures in commonly used statistical software such as SAS (SAS, 2000) and SPSS (SPSS, 2002). Pairwise data deletion is also available in a number of SAS and SPSS statistical procedures.

3.3.1.3 Re-weighting

Re-weighting is another technique that has been used to handle missing values, especially *unit* nonresponse in large national surveys (Little and Rubin, 1987; Little and Vartivarian, 2003). The idea is to discard incomplete cases and reweight the complete ones so that they more closely resemble the population with respect to distribution of important characteristics. The simplest form of the weighting

approach is *complete-case analysis*, where the complete cases are all given the same nonresponse weight. Typically, cases with all values present receive higher weights to counterbalance cases with missing data (Little and Rubin, 1990). Nonresponse weighting increases the weight of complete cases to represent the entire sample irrespective of missingness. Weighting can be a useful tool to reduce bias which arises from restricting analyses to complete cases and when missingness is not missing completely at random (Schafer and Olsen, 1998). Despite its simplicity and the advantage of correcting biases due to differential response that is related to the variables used in the adjustment and not requiring models for data, weighting has the limitation of being strictly applicable to only monotone patterns of data. The technique is also inefficient because the exclusion of observed data from partially complete observations reduces sample size. The derivation of appropriate standard errors from the weighted analysis can also be a difficult task.

3.3.2 Imputation Techniques

Imputation has become one of the most popular and useful tools used to solve missing value problems in survey data analysis, especially *item* nonresponse (or *partial* nonresponse). There are several ways to "impute" missing values, most of which are based on statistical procedures. The attraction of imputation is that once the missing data are filled-in (imputed), all the statistical tools available for the complete data may be applied.

Different categorisations of imputation can be distinguished. First, imputations may be deterministic or random (stochastic). In the first case, imputations are determined by the incomplete data table, and are the same if the method is applied again. In the latter case, imputations are randomly drawn either from observed data or from a predicted distribution. A second distinction is that between naïve and more principled approaches. Naïve methods are quick options mainly based on analysing complete cases whereas more principled approaches use models for both the observed and missing data on which the imputations are based. Finally, imputation may be based on explicit and implicit models (Little and Schenker, 1995). Explicit models are the type of models usually discussed in mathematical statistics, for

example, normal linear regression models. Implicit models are models which underlie procedures for fixing up data structures in practice and are often nonparametric. Different kinds of information could be used to impute missing values.

Most imputation procedures for missing data, including maximum likelihood methods, are single imputation. This is probably the most common method for handling item nonresponse in current survey practice. Such single imputation procedures are further described.

3.3.2.1 Single Imputation Techniques

One approach to dealing with missing values is single imputation. Single imputation refers to filling in a missing value with a single replacement value. There are two general approaches: arbitrary methods and regression-based imputation. Different kinds of information may be used to impute missing data. These are discussed below.

3.3.2.1.1 Mean or mode imputation

Some researchers have used arbitrary methods like mean imputation (apparently first mentioned by Wilks (1932)) for addressing the missing value problem, i.e., replacing the missing values of a variable by the mean of its observed values. Mean substitution also assumes a MCAR mechanism. The strength of mean imputation is that it preserves the data and it is easy to use. However, mean imputation can be misleading because it produces biased and inconsistent estimates of both coefficients and standard errors (Tresp *et al.*, 1994). Little and Rubin (1997) points out that variance parameter estimates under mean imputation are generally negatively biased. Also, substitution of the simple (grand) mean will reduce the variance of the variable and its correlation with other variables. Somewhat better is substitution of the group or global mean (or mode in the case of nominal data) for a grouping variable known to correlate as highly as possible with the variable which has missing values. We shall call this technique mean or mode single imputation (MMSI).

3.3.2.1.2 Hot deck imputation

Non-parametric imputation methods like "hot deck" imputation (Ford, 1983; Sande, 1983) include replacing the missing value by a value observed in the data set. The main principle of the hot deck method is using the current data (donors) to provide imputed values for records with missing values. All observations are divided into groups or classes with similar characteristics, i.e., identify the most similar case to the case with a missing value and substitute the most similar case's, say, Y value for the missing case's Y value. The procedure through which we find the donor that matches the instance with missing values is different according to the particular technique used. Creating a larger number of subgroups yield some improvement in accuracy, but it can also lead to a very small sample size within some subgroups.

One advantage of hot decking is that, unlike arbitrary methods, it reflects both the mean and variance of the underlying data. Other advantages of hot decking is that it preserves the distribution of item values; permits the use of the same sample weight for all items; and the results obtained from different analyses are consistent with one another. The primary drawbacks of this method are the lack of guidance in creating the subgroups and the possibility of creating subgroups with few observations. Ernst (1980) found that in general hot deck procedures led to higher variances but reductions in biasness. Also, hot deck imputation tends to be robust, especially for small data sets. Hot deck imputation methods include sequential, hierarchical, multivariate matching, record matching, predictive mean matching, distance function matching or nearest neighbour imputation in which a nonrespondent is assigned the item value of the nearest neighbour. Non-invasive imputation (a procedure based on non-numeric rule based data analysis), which aims to maximise consistency of imputation in known values) is another variation of hot deck (Gediga and Düntsch, 2003). The approach is non-invasive because it takes all its information from the given data and makes no additional dependency or distributional assumptions. When historical or older data is used the approach is cold deck imputation. Cold deck imputation is appropriate for the imputation of panel surveys (where people are interviewed several times). Also, cold deck is useful for variables that are static, such as place of birth or gender.

3.3.2.1.3 Regression-based imputation

Regression-based imputation methods have also been used for handling missing data by Buck (1960) and Afifi and Elashoff (1966). This technique is similar to hot-decking, except it is somewhat more flexible. This type of technique estimates (imputes) missing data values based on other variables in the data set. Regression imputation comes in various forms. One form is the multiple regression strategy. For this technique, one develops a regression equation based on complete case data for a given variable, treating it as the outcome and using multiple other relevant variables as predictors. Then, for cases where, say, Y is missing, plug the available data into the regression equation as predictors and substitute the equation's predicted Y value into the database for use in other analyses.

The proper regression model depends on the form of the dependent variable. A probit or logit is used for binary variables, Poisson or other count models for integer-values variables, and ordinary least squares (OLS) or related models for continuous variables. This method can be often unsatisfactory for nonlinear data and biased if modal misspecification occurs. Another shortcoming of applying estimation methods is that variability in the imputed values is underestimated in comparison with variability of non-imputed values, i.e., the uncertainty due to the missing data is not taken into account with similar implications of standard errors of model parameters. To solve this problem a random error term is added to the value fitted by the regression estimator. The combination of regression-based imputation with the addition of a random error term is known as stochastic (regression) imputation.

Another form of regression imputation is the stepwise or iterative regression approaches. When using stepwise regression, only the key variables that contribute to imputation are isolated. The iterative approach requires computation of an initial correlation matrix, which is then used to compute some regression equations. The new missing values are imputed and substituted in the data matrix. This process is repeated until there is very little change noted in the regression weights.

3.3.2.1.4 Expectation maximization

Maximum Likelihood (ML) estimation is another approach to analysing missing data by using all available data points in a database to construct the best possible first and second order moment estimates under the missing at random (MAR) assumption. ML methods are model-based. That is, they are implemented as part of a fitted statistical model.

A very closely related method to ML is the classical Expectation Maximization (EM) algorithm (Dempster *et al.*, 1977; Little and Rubin, 1987) that has been used for model-based imputation. EM is an iterative regression technique in which missing variables are regressed on the available data and any additional variables provided as inputs to the algorithm. First, a vector of means and covariance matrix are calculated using all available data. The means are then imputed for missing values in each variable. These imputed means serve as a starting value for the imputation. Next, variables with missing values are regressed on all the other available variables, and a residual term is added to each missing value to correct for random variability lost in the imputation process. Naturally, a different regression is performed for each pattern of missing data. The imputed values are then replaced with estimates calculated from the regression equations. With the new imputations in place, the means and covariances are recalculated. Regression equations and imputations are iteratively calculated until the mean and covariance matrix values converge (Wu, 1983; Schafer, 1997; Allison, 2001). In other words, the EM algorithm is typically used to make imputations about missing data by capitalising on the relationship between missing data and the unknown parameters of a data model.

There are two main applications of the EM algorithm. The first occurs when the data has missing values. The other occurs in likelihood inference with mixed models, whereby the likelihood function involves an analytically intractable integral with respect to the random effects distribution. The latter application is more common in the computational pattern recognition community.

Theoretical Derivation of EM algorithm is given below:

The EM algorithm is a general computational method of calculating maximum-likelihood estimates through a two-step iteration: **E**xpectation and **M**aximization. Since it is simple and stable, the EM has been widely used to fit models from incomplete data.

Given some data $X = \{x_1,..., x_N\}$ and a model family parameterised by θ, the goal of EM is to find θ such that the likelihood $L(X \mid \theta)$ is maximised. We shall not be giving the general description of the EM but of a special case, i.e., the EM for mixtures. First, we find the maximum likelihood parameters of a mixture model (Dempster *et al.*, 1977; Everitt and Hand, 1981; McLachlan and Basford, 1988), assuming that the data X is generated independently from a mixture density

$$f(x) = \sum_i f(x \mid c_i; \theta_i)P(c_i) \tag{3.1}$$

where each component of the mixture is denoted by c_i. From equation 3.1 we can define the log likelihood function as:

$$L(\theta \mid x) = \sum_j \log f(x_j) = \sum_j \log \sum_i f(x_j \mid c_i; \theta_i)P(c_i) \tag{3.2}$$

which is hard to solve because of the log of a sum. The EM could be used for solving this problem.

The core idea of the EM algorithm is to introduce some unobserved variables $\boldsymbol{Z}$, appropriate for the model under consideration, such that if $\boldsymbol{Z}$ were known the optimal value of θ could be computed easily. Then the complete conditional probability density (including the missing variables) can be written as:

$$L(\theta \mid X, Z) = \sum_{i=1}^{N} \sum_{j=1}^{M} z_{ij} \log f(x_i \mid z_i; \theta) f(z_i; \theta) \tag{3.3}$$

The usual approach is to regard Z as missing data and estimate it iteratively.

The intuition behind the EM algorithm is that we would like to maximise the complete data likelihood but it cannot be utilised directly, so we maximise its expectation, denoted by $Q(\theta \mid \theta^t)$, instead. As shown by Dempster *et al.* (1977),

$L(\theta \mid X)$, the complete data likelihood can be maximised by iterating the following steps:

1. Initialise parameters randomly. Set $t = 0$.
2. *E-step*: Determine $Q(\theta \mid \theta^{(t)}) = E[L(\theta \mid X, Z) \mid X, \theta^{(t)})]$
3. *M-step*: Set $\theta^{(t+1)} = \arg\max_{\theta} \{Q(\theta \mid \theta^{(t)})\}$

   where $\theta^{(t)}$ are the current parameter estimates in time step t.
4. Iterate steps 2 and 3 until convergence

Assume that the data set $X = \{x_1, ..., x_N\}$ is divided into an observed X_{obs} and missing X_{miss} components, respectively. To handle missing values we can re-write the EM algorithm as follows:

1. Initialise parameters randomly. Set $t = 0$
2. *E-step*: Determine $Q(\theta \mid \theta^{(t)}) = E[L(\theta \mid X_{obs}, X_{miss}, Z) \mid X_{obs}, \theta^{(t)})]$
3. *M-step*: Set $\theta^{(t+1)} = \arg\max_{\theta} \{Q(\theta \mid \theta^{(t)})\}$ where $\theta^{(t)}$ are the current parameter estimates in time step t.
4. Iterate steps 2 and 3 until convergence

The expectation (E) step computes the expected values for the sufficient statistics given a model and values for model parameters θ, i.e., the expected value of the complete data likelihood with respect to the missing data given the observed data and the current parameter estimates. The maximisation (M) step estimates the model parameters by maximising the likelihood using standard procedures, given complete data. The procedure iterates through these two steps until convergence is obtained. Convergence occurs when the change in parameter estimates from iteration becomes negligible. An important part of the EM algorithm is restoring error variability to the imputed values during the E-step. The SPSS Missing Values Analysis (MVA) module employs the EM approach to missing data handling (SPSS, 1997).

The strength of the EM approach is that it has well-known statistical properties and it generally outperforms popular *ad hoc* methods of incomplete data handling such as listwise and pairwise deletion and mean substitution because it assumes incomplete cases have data missing at random rather than missing completely at random. Also, EM-imputations ignore any estimation error for the missing data, which will, in turn, lead to negatively biased standard error estimates in any model you run on complete data, overly significant *p*-values, variance parameter estimates may also show some negative bias, and so on. Rubin (1987) calls this type of imputation "improper" because the method does not adjust for the fact that the mean squared error and the parameters used to produce the predicted values for all cases including those with data missing are only estimates, not the true values.

The biggest drawback with EM is that it typically does not provide standard errors (and confidence intervals) as a by-product of the parameter estimation. Thus, although the parameter estimation itself is excellent with EM, it is not possible to do hypothesis testing with the EM-based estimates unless one does a separate step specifically for that purpose, such as bootstrapping (Efron, 1982). To use EM the algorithm we have to specify the sample distribution in advance. Unfortunately, in many data mining problems we do not know the probability density function in advance. The EM algorithm has a linear convergence determined by the rates or fraction of missing information in the data set. When the fraction of missing values is large with one or more parameters missing, then convergence will require many iterations and thus will be very slow. Furthermore, the material explaining the EM algorithm is complex and requires a high level of technical expertise in the use of programs.

3.3.2.1.5 Full information maximum likelihood

The method of full information maximum likelihood (FIML), which is also known as raw maximum likelihood, is another theory-based approach to the treatment of missing data. Hartley and Hocking (1971) did the original work of FIML to cope with missing data. The FIML approach uses maximum likelihood estimation for incomplete data. FIML assumes multivariate normality, and maximises the

likelihood of the model given the observed data. This assumption implies two things: All variables have normal distributions. Secondly, each variable can be represented as a linear function of all the other variables, together with a normal, homescedastic error term (Allison, 2001). All available data is used to generate maximum likelihood sufficient statistics. Usually these consist of a covariance matrix of the variables and a vector of means.

It is important to note EM and FIML are equivalent, i.e., they both give ML estimates of the covariance matrix, but simply do so using different numeric algorithms. However, unlike the EM approach, FIML allows for the direct computation of appropriate standard errors and test statistics. These estimates of standard errors (and confidence intervals) of regression model parameters are not provided in EM and without additional analytic steps such s bootstrapping. As noted by Graham *et al.*, 1997, standard errors would be provided on computer output when using the EM covariance matrix as input for further analyses (for example, multiple regression), but these standard errors would be based on the wrong sample size and thus are incorrect. Also, unlike EM, FIML can be employed in the context of user-specified linear models, such as structural equation models, regression models, Analysis of Variance (ANOVA) and Analysis of Covariance (ANCOVA) models.

FIML approach has the advantage of convenience or ease of use and well-known statistical properties. It also produces unbiased parameter estimates and standard errors under MCAR and MAR (Arbuckle, 1996a). Limitations of the FIML approach include an assumption of joint multivariate normality of variables used in the analysis and the lack of a raw data matrix produced by the analysis. Also, this approach assumes that data are MAR. FIML is currently implemented in Mx (Neale, 1994) and AMOS (Arbuckle, 1996b; Byrne, 2001) structural equation modelling packages. Other software packages that use the FIML approach to handle incomplete data are the MIXED procedure in SAS (Latour *et al.*, 1994). It is also important to note that the FIML estimator does not impute or fill-in missing values but directly estimates model parameters and standard errors using all available raw data.

3.3.2.1.6 Other single imputation techniques

When a categorical variable has missing values it is common practice to add an extra "missing value" category. However, this could be bad practice because the impact of this strategy depends on how missing values are divided among the real categories, and how the probability of a value being missing depends on other variables. Also, very dissimilar classes could be lumped into one group.

Other data imputation techniques are used to replace the missing observations with plausible values other than the mean, and then analyse the "complete" data set. These include: using the most common value (mode), series mean, the mean or median of nearby points, or linear interpolation between prior and subsequent known points, or substitution of the linear trend value for that point. In some other cases, the imputed values are based on previously observed values. This method is referred to as Last-Value-Carried-Forward (LVCF) or Last-Observation-Carried-Forward (LOCF) technique which is based on a very strong assumption of stability. This method works best if the observation is expected to remain at some level or if there are only a few missing values. However, this technique can only be used for longitudinal variables with multiple time-points. Other arbitrary methods can be created as well.

The main goal of all single imputation is to achieve complete data wherever possible so that they can be analysed by methods designed for complete data. However, the choice of technique is problem dependent. One of the strengths of single imputation is that standard complete-data methods can be used once the missing values have been imputed. Also, the single imputations created by the data collector can incorporate their knowledge which could prove advantageous to the unsophisticated end user who has to analyse the missing data. Despite its strengths, single imputation has drawbacks. Imputation methods can be distinguished as model based or parametric as opposed to being data based or non-parametric. Parametric imputation methods include substituting the sample mean values for the available cases for the missing values. This is an attractive idea but it has its own pitfalls. The main one is that whenever the missing data are replaced by one set of imputed

values, later analyses will not reflect missing-data uncertainty, and thus overstate precision, i.e., an analysis which ignores the uncertainty of missing data prediction will lead to the sample size being overestimated, standard errors that are too small, p-values that are artificially low, and rates of Type I error that are higher than nominal level. Furthermore, when nonresponse is not really understood, no account is being taken of uncertainty arising from not knowing which nonresponse models for imputation are appropriate.

3.3.2.2 Multiple Imputation

A serious defect with imputation is that it seems to be inventing data. More specifically, a single imputed value cannot represent all the uncertainty about which value to impute, so the analyses that treat the imputed values just like observed values generally underestimate uncertainty, even if nonresponse is modelled correctly and random imputations are created. A replication method known as multiple imputation (MI) is designed to address the inherent weaknesses of single imputation while retaining their advantages (Rubin, 1976; Rubin and Schenker, 1986; Little and Rubin, 1987; Schafer and Olsen, 1998; Allison, 2001; Enders, 2001). Instead of imputing a single set of draws from the missing values, multiple random draws are simulated from the population. One approach of this simulation is to use bootstrap methods. Creating multiple bootstrap datasets would (to an extent) be like taking multiple draws from the population. Another approach is to simulate these random draws with data augmentation (Tanner and Wong, 1987). The art of data augmentation (DA) is discussed below.

DA is one iterative regression based or simulation-based approach that has strong similarities in many ways to an EM approach. DA (Tanner and Wong, 1987; Schafer, 1997) has also been used to great advantage by the EM algorithm in solving maximum likelihood problems. Both approaches view the observed data of a statistical model as incomplete, augmenting the missing data, and making inferences about the unknown parameters. However, DA does this in a stochastic or random fashion. The result of DA is a predictive distribution of missing values that is used to impute missing values.

DA follows the following process:

1. Initialise parameters randomly. Set $t = 0$
2. *I-step*: Given a current estimate $\theta^{(t)}$, select a value of the missing data from the conditional predictive distribution of $X_{miss}, X_{miss}^{(t+1)} \sim P(X_{miss} \mid X_{obs}, \theta^{(t)})$
3. *P-step*: Conditioning on $X_{miss}^{(t+1)}$, draw a new value of θ from its complete data posterior, $\theta^{(t+1)} \sim P(\theta \mid X_{obs}, X_{miss}^{(t+1)})$. Through an iterative process two distributions are obtained, $P(\theta \mid X_{obs})$ and $P(X_{miss} \mid X_{obs})$. For a suitable large t, we can implement a DA algorithm by Tanner and Wong (1987), which iterates between sampling θ^{t+1} from $P(\theta \mid X_{obs})$ and sampling $X_{miss}^{(t)}$ from $P(X_{miss} \mid X_{obs})$.
4. Iterate steps 2 and 3 until convergence

The Imputation (*I*) step simulates a random imputation of missing data under assumed values of the parameters. The Posterior (*P*) step draws new parameters from a Bayesian posterior distribution based on the observed and imputed data. The procedure of alternately simulating data and parameters creates a Markov Chain (MC) $X_{miss}^{(1)}, \theta^{(1)}, X_{miss}^{(2)}, \theta^{(2)}, \ldots$ (Gilks *et al.*, 1996), which eventually stabilises or converges in distribution to $P(X_{miss}, \theta \mid X_{obs})$. The procedure iterates through these two steps until convergence is obtained. The rate of convergence is related to the fraction of missing information. DA can be thought of a small–sample refinement of the EM algorithm using simulation, with the imputation step corresponding to the E-step and the posterior step corresponding to the M-step.

Despite its desirable properties, DA has several limitations. First, it is only a local and deterministic maximiser of the likelihood and its asymptotic behaviour depends heavily on the starting values or initial conditions. More seriously, apart from some simple models, the algorithm is far from easy to set up: namely the *I*-step as well as

the *P*-step - or both steps in even worse situations - can be intractable or numerically inefficient. Also, alternating between these two steps to set up a Markov chain that converges to a stationary distribution, the joint distribution of the missing data and parameters given the observed data have heavy computational requirements.

The EM and DA approaches can be combined and used to generate imputations. The parameter estimates from EM provide convenient starting values for DA. Moreover, the convergence behavior of EM provides useful information on likely convergence behavior of DA. If EM converges within a certain number of iterations, say 50, then in nearly all situations it will be sufficient to allow 50 steps of DA between successive imputations. There are two ways to calculate multiple imputations after the k intervals of one long chain, or saving the final imputations from several parallel chains using different starting values. Gelman and Rubin (1992) recommended using the second approach and in doing so to use starting points which are overdispersed relative to the observed data posterior using the starting values.

MI is similar to the ML method except that MI generates actual raw data values suitable for filling in gaps in an existing data base. Instead of filling in a single value for each missing value, the MI procedure replaces each missing value with a set of plausible values that represent the uncertainty about the right value to impute. These multiple imputed data sets are then analysed by using standard procedures for complete data and combining the results from these analyses into a single summary finding. This results in statistically valid inferences that properly reflect the uncertainty due to missing values. In other words, when multiple imputations represent repeated random draws under one model for nonresponse, valid inferences that reflect the additional variability due to the missing values under that model are obtained in a straightforward manner. MI has two more advantages. First, when imputations are randomly drawn in an attempt to represent the distribution of data, MI increases the efficiency of the estimation. Finally, when repeated randomly drawn imputations are created under more than one model, MI facilitates the straightforward analysis of the sensitivity of inferences to various models for nonresponse simply by repeatedly using complete-data methods (Rubin, 1987).

MI has several desirable features: 1) Introducing appropriate random error term into the imputation process which makes it possible for the method to get approximately unbiased estimates of all parameters, 2) Repeated imputation allows one to get good estimates of standard errors; 3.) MI can be used with any kind of data and any kind of analysis without specialized software, 4.) MI saves money, since for the same statistical power, MI requires a smaller sample size than, say, listwise deletion, and 5) Once imputations have been generated by a knowledgeable user, researchers can use them for their own statistical analysis. However, certain requirements must be met for MI to have these desirable features. First, the data must be MAR. Second, the model used to generate the imputed values must be 'correct' in some sense. Lastly, the model used for the analysis must match up, in some sense, with the model used in the imputation. The reader is referred to (Schafer, 1997; Allison 2001) for a rigorous description of all these conditions.

This approach shall now be called Expectation-Maximization multiple imputation (EMMI). However, if a missing value is replaced with only a single value, the approach shall be called Expectation-Maximization single imputation (EMSI).

3.4 Decision Trees and Missing Data

Several methods have been proposed in the literature to treat missing data. Missing values can cause problems at two points when using decision tress; 1) when deciding on a splitting point (when growing the tree), and 2) when deciding into which daughter node each instance goes (when classifying an unknown instance).

Methods for taking advantage of unlabelled classes can also be developed, although we do not deal with them in this thesis, i.e., we are assuming that the class labels are not missing.

Specific decision tree techniques for handling missing data are now going to be discussed. These MDTs are divided into three categories: ignoring and discarding data, imputation and machine learning. However, most of the techniques under these categories have already been discussed in Section 3.4.1. Thus, they are not covered in detail in this section.

3.4.1 Imputation Techniques

3.4.1.1 Single Imputation Techniques

3.4.1.1.1 Mean or mode imputation

MMSI has been used for handling incomplete data when using DTs; it consists of replacing the missing attribute values by the means (for continuous attributes) or the modal value or most common value (for nominal attributes). This is a strategy whereby you replace the unknown values with the most common values for the attribute found in the training set. The decision tree is then induced from the completed data. This approach is based on the assumption that all "missings" are somehow typical.

3.4.1.1.2 Conditioning on class imputation

Kononenko and Roscar (1984) follow a MMSI approach by conditioning that particular attribute with missing value(s) to the class associated with the missing value. For a continuous attribute, the mean value, $\overline{a}_i$, of attribute A of the missing value, given the class of the instance concerned is used. For a nominal attribute, the most common value (mode), $\overline{a}_i$, of attribute A is used to estimate the missing value. This method shall now be called conditioning on class single imputation (CCSI). CCSI has the limitation of being applicable only in the training case, where the class variable is present and not in the testing case. The impact of this strategy also depends on how missing values are divided among the classes.

3.4.1.1.3 "New" categorical value

This is an approach followed by Quinlan (1979; 1986; 1987), which models the probability of missingness. "Missing" is treated as a distinct splitting value. In this approach, if a missing value occurred at a nominal attribute that is used for branching, it creates a new branch called unknown. For an ordinal attribute, the missing values constitute a special value that is assigned in the ordering that yields the best split. The place is generally different in different nodes of the tree. This

strategy is normally used when building a decision tree with incomplete data. However, it could still be used for classification tasks.

This method has two advantages: no instances are dropped due to the missing values, and unobserved similarities among instances with missing values will be captured by the new term. The impact of this strategy depends on how missing values are divided among the real categories, and how the probability of a value being missing depends on other variables. In addition, treating "missing" as a separate level is a good idea when the "missing" is informative and there are a lot of instances in the training set. Missing values normally follow some basic missing data mechanism. These mechanisms have already been discussed in Section 3.2. Most of the time missing values are missing at random, and in these circumstances, the new value ("unknown") would not have the same importance as a real attribute value. Also, this method works well for categorical attributes but continuous attributes cannot be modelled in this way without being discretized first before applying learning algorithms (like DTs) to datasets. In addition, working with discretized attributes often produces better results, or even work faster (Kebber, 1992; Fayyad and Irani, 1993; Frank an Witten, 1999). The strategy also works well when missing values are indicative of certain target values. For example, people with large incomes might be more reluctant to disclose their income than people with ordinary incomes. If income were predictive of a target, then missing income would be predictive of the target, and the missing values would be regarded as a special large income value. Also, the strategy seems harmless when the distribution of missing values is uncorrelated with the target because no choice of branch for the missing values would help predict the target.

When two instances that have the same field empty are compared for equality, it is not always clear whether the result of comparison for that attribute should be true or missing. Hence, treating a missing value as an additional category for each attribute can cause problems during analysis. If there is a missing value for an attribute, a special value "unknown" is used as just another or additional new value of that attribute and dealt with in the same way as other values. Hence, the number of values is increased by one for each attribute that depicts the unknown value in

the training set. This can be a problem for methods that do multi-way splits due to the increase in the number of levels. (Quinlan, 1986) suggests using only binary splits to outwit this problem. This approach is also based on the assumption that whatever the unknown value, it is the same for all cases with missing values. This could be a problem as there can be more than one reason for a database field to be missing.

3.4.1.1.4 Attribute value matching imputation

Bruha and Franek (1996) suggest matching complexes with instances that involve unknown attribute values, both in learning and classification. Bruha and Franek follow the most common value approach for both learning and classification purposes using class-sensitive absolute frequencies and overall frequencies, respectively. An unknown value V of an attribute A of an instance belonging to class C is replaced by the class-sensitive common value which maximises the Laplacian formula $(N_{r,j,n}+1)/(N_{n,j}+R)$ over j for a given r and n where $N_{r,j,n}$ is the number of instances belonging to class C_r exhibiting the value V_j for each attribute value A_n; $N_{n,j}$ is the number of instances exhibiting the value V_j for each attribute value A_n; R is the number of classes. The Laplacian criterion is used for expected accuracy for the class C_r. If the maximum is reached for more than one value of the attribute then the value with the greatest frequency $N_{r,j,n}$ is selected as the common value. For testing, the unknown value is replaced by the overall common value which maximises $N_{n,j}$ over the subscript j. This approach is appropriate for handling categorical attributes with missing values.

3.4.1.1.5 All possible values imputation

Grzymala-Busse and Hu (2000)'s cautious imputation technique is similar to Quinlan (1989) but in this method, an instance with a missing attribute value is replaced by a set of new instances, in which the missing attribute value is replaced by all possible values of the attribute. For example, if attribute A has a missing value for instance I, and attribute A has r possible values, then I will be replaced by r new instances $I', I'', ..., I^{(m)}$. When instance I has two unknown values of attributes A_1 and A_2 and there are r possible values of A_1 and c possible values of A_2, then I

will be replaced by r x c examples and so on. The rationale of the method is that since the value of an attribute A for a given instance I is missing, every possible value of A is considered, and every such value corresponds to a new instance. This method produces inconsistent decision tables. The decision table is inconsistent when it contains at least one pair of inconsistent instances, i.e., instances characterised by the same values of all attributes yet with different values of a decision. However, the problem of inconsistent decision tables is solved using rough set theory (Pawlak, 1991), by producing two sets of rules: certain and possible. Certain rules are categorical, while possible rules are supported by existing data, although conflicting data may exist as well. For possible rules an estimate for the worst case of error is presented. The presented approach may be combined with any other approach to uncertainty when processing of possible rules is concerned.

3.4.1.1.6 Unordered decision tree imputation

The unordered attribute decision trees, which can also be considered a machine learning technique as it uses a DT algorithm to impute missing values, is another strategy that has been used for handling missing values in tree learning. This technique was suggested by Shapiro (1987) and followed up by Quinlan (1987). The method builds decision trees to determine the missing values of each attribute, and then fills the missing values of each attribute by using its corresponding tree. The strategy shall now be referred to as decision tree single imputation (DTSI). Separate trees are built using a reduced training set for each attribute, i.e., restricting your analysis to only those instances that have known values. Hence, as many decision trees as the attributes in the domain are constructed. The original class is treated as another attribute, while the value of the attribute becomes the "class" to be determined. The attributes used to grow the respective trees are unordered. These trees are then used to determine the unknown values of that particular attribute. The grown tree can then be used to classify a new instance in the reduced set with the unknown value of that particular attribute determined. Although this method could be more exact in filling in missing information, it significantly increases the computational cost. Also, this method makes sense when building a decision tree using only categorical attributes whereby a classification tree is used to estimate the

missing attribute values. For non-categorical attributes a regression tree could be used instead. The approach is also suitable for domains in which strong relation between attributes exist.

3.4.1.1.7 Ordered decision tree imputation

Lobo and Numao (1999; 2000) follows-up Quinlan's DTSI approach but by first ordering the attributes using mutual information before growing the tree. As with Quinlan's method, only those attributes with known values and low mutual information with respect to class are included in the reduced training set. After constructing a decision tree for filling the missing values of an attribute, it makes sense to use the data with filled values in order to construct a decision tree for filling the missing values of other attributes. Thus, the order followed when constructing attribute trees and filling the missing values per attribute becomes important. The special ordering on the attribute's trees construction was empirically found to improve the accuracy of the decision tree learning algorithm, while keeping the computational cost to a sustainable level (Lobo and Numao, 1999). Even though this technique makes good use of all the information (the class and the attribute variables take part in the estimation), it has a weakness of not performing well if the same case has missing values in more than one attribute.

Single imputation is more efficient than instance deletion, particularly for item nonresponse. However, it requires care to avoid data distortion. It also has the drawback of not taking into account missing-data uncertainty when estimating the missing values. The application of standard complete-data methods to single imputed data set treats the missing values as if they were known (Schafer, 1997). The Full Information Maximum Likelihood (FIML) and Multiple Imputation (MI) methods account for the uncertainty in estimating the missing data.

3.4.1.1.8 Bayesian imputation

The main idea of this strategy relies on a probabilistic framework and was proposed by Cestnik *et al.*, (1987). This more complex procedure assigns a probability to each of the possible values of attribute A rather than simply assigning the most common value to $A(x)$. These probabilities are estimated based on the observed frequencies of

the various values for A among the examples at node n. The method works as follows:

A model for the missing values is constructed. When learning the tree, *estimation* of the conditional probabilities of right and left splits (of the missing value) given all the observed information is used. Each example is split into a probability distribution over leaves and estimated from the relative frequencies of the attribute values among the training instances collected at the node.

The most usual thing to do is to associate an instance with a missing value for the given node to all branches, weighted with the conditional probability of the corresponding value given the class (for training instances) or unconditional probability of the corresponding value (for testing instances). This is done as follows.

Suppose that the given instance (with the unknown value) belongs to a class C. Then the probability of an attribute A having a value V is:

$$P(V \mid C) = \frac{P(V \,\&\, C)}{P(C)} \tag{3.12}$$

where the calculation of $P(V \,\&\, C)$ and $P(C)$ are approximated with relative frequencies from the set of instances in the current node of the tree.

When training a tree, a training instance with an unknown value for an attribute A is split into a set of examples so that we have for each possible value V of A one example weighted with the probability $P(V \mid C)$.

With a testing instance you do not know the value of class. So, an object with a missing value of an attribute A is classified by following the branches that correspond to all possible values of A, weighted by the prior probabilities $P(V)$ of the corresponding values of A. All the branches for all values of the attribute A are followed and the final decision (classification) is the class with the highest probability. The method works well for categorical data but does not perform well with non-categorical data.

3.4.1.2 Multiple Imputation

MI is a simulation approach to missing data that works with standard complete-data analysis methods. MI means that the missing data are imputed a number of times, typically 3 to 5 times, with a different randomly chosen error term added in each imputation. In order to create imputed value we need to identify some model, which will allow us to create values ("imputes") based on other variables in the data set. By imputing missing values several times, a few augmented data sets are created such that regular complete-case analyses can be easily performed. The completed data sets are analysed using standard methods and the results are combined using rules established by Rubin (1987). These rules allow the analyst to produce one set of estimates like that produced in non-imputation analysis. The parameters of interest, then, can be calculated by averaging the parameter estimators from each augmented data set.

There are many implementations of MI. An excellent option is Schafer's (1997) set pf programs headed by the NORM program for MIL under the multivariate normal model. NORM provides a parametric technique that uses a Bayesian procedure known as 'data augmentation' to iterate between random imputations under a specified set of parameter values and random draws from the posterior distribution of the parameters (given the observed and imputed data). NORM assumes that the data come from a multivariate normal distribution and are missing at random. The program work with continuous data but it has been shown to perform well with categorical data. Schafer (1997) has three other MI programs. PAN is available for special longitudinal panel data situations and cluster data when there are many clusters. CAT uses a saturated multinomial model and a constrained loglinear model (Bishop *et al.*, 1975) to impute categorical variables; MIX uses restricted and unrestricted general location models (Olkin *et al.*, 1961) to impute mixed variables (include both categorical and continuous variables in one model). Details about these models can be found in Schafer (1997).

There is also another Bayesian approach to imputation which has been presented by Chiu and Sedransk (1986). The method makes use of the EM algorithm and is

similar to the MI procedure proposed by Rubin (1987). Its advantage lie in the fact that it uses standard statistical procedures, does not require model assumptions, permits a general specification of the response mechanism, and allows the input prior information in a routine manner. The method is best suited when substantial differences are expected between respondents and nonrespondents.

3.4.2 Machine Learning Techniques

Machine learning (ML) techniques are those that deal with missing values using machine learning algorithms. ML techniques are generally more complex than statistical techniques. This section will describe missing data imputation using two supervised machine learning systems: surrogate variable splitting, fractioning of cases and dynamic path generation. The rest of the section will describe other methods that have been used for handling incomplete data using DTs.

3.4.2.1 Surrogate Variable Splitting

The greatest advantage of CART models is their ability to deal with missing values. One sophisticated but refined method worthy of note and study is the surrogate variable splitting (SVS), which has been used for the CART system and further pursued by Therneau and Atkinson (1997) in RPART. CART handles missing values in the database by substituting "surrogate splitters". Surrogate splitters are predictor variables that are not as good at splitting a group as the primary splitter but which yield similar splitting results; they mimic the splits produced by the primary splitter; the second does second best, and so on. The surrogate splitter contains information that is typically similar to that which would be found in the primary splitter. The surrogates are used for tree nodes when there are values missing. The surrogate splitter contains information that is typically similar to what would be found in the primary splitter. Both values for the dependent variable (response) and at least one of the independent variables (attributes) take part in the modelling. The surrogate variable used is the one that has the highest correlation with the original attribute (observed variable most similar to the missing variable or a variable other than the optimal one that best predicts the optimal split). The surrogates are ranked. Any observation missing on the split variable is then

classified using the first surrogate variable, or if missing that, the second is used, and so on. The CART system only handles missing values in the testing case but RPART handles them on both the training and testing cases.

The basic idea is as follows:

When building a tree with incomplete vectors, at node t, find the best split s_m^* on the feature instance x_m using all the training samples containing a value of x_m. Then select the split s^* which maximises the impurity reduction $\Delta i(s_m^*, t)$ at node *t* (See Section 2.2.1). In other words, you choose a primary predictor and split point. A primary splitter is the best splitter of a node. Then, you compose a list of surrogates (surrogate predictors) and split points; surrogate splitters mimic the splits produced by the primary splitter. The optimal split point for a surrogate maximizes the association between the surrogate and the primary splitter.

When classifying a new instance, if at node *t* the best split s^* is not defined because of missing feature instances, proceed as follows. Examine all non-missing feature instances for the test sample; find that feature instance, say x_m, with split $\tilde{s}_m$, which is most similar to s^*. Therefore, $\tilde{s}_m$ is called a surrogate split of s^*. Finally, use $\tilde{s}_m$ at node *t* to decide to traverse to node t_L or t_R. In other words, when sending observations down the tree, use the primary predictor split first. If the value of the primary split is missing, use the first surrogate. If the first surrogate is missing, use the second, and so on. If an instance is missing all the surrogates the blind or majority rule is used.

Surrogate splitters are similar to competitor splitters in the sense that they both yield splits of benefit but are not as good as the primary splitter. Often, the same variable will be listed as both a competitor and a surrogate. However, there is a significant difference between the way variables are ranked as competitors and as surrogates. Competitor splits are runners-up to the primary split: they are judged the same way the primary splitter is judged by how much improvement they make in reducing node impurity. Surrogate splitters are not ranked by the amount of improvement they produce but rather by how closely mimic the split selected for the

primary splitter. The optimal split point for a surrogate maximizes the association between the surrogate and the primary splitter; it does not necessarily maximize the improvement. If you compare entries for the same variable in the competitor and surrogate lists, you may see different split points selected and different values for the improvement from the splits.

The idea of surrogate splits is conceptually excellent. Not only does it solve the problem of missing values but it can help identify the nodes where masking or disguise (when one attribute hides the importance of another attribute) of specific attributes occurs. This is due to its ability to making use of all the available data, i.e., involving all the attributes when there is any observation missing the split attribute. By using surrogates, CART handles each instance individually, providing a far more accurate analysis. Also, other incomplete data techniques treat all instances with missing values as if the instances all had the same unknown value; with that technique all such "missings" are assigned to the same bin. For surrogate splitting, each instance is processed using data specific to that instance; and this allows instances with different data patterns to be handled differently, which results in a better characterisation of the data (Breiman *et al.*, 1984).

However, practical difficulties can affect the way surrogate splitting is implemented. Surrogate splitting ignores the quantity of missing values. For example, a variable taking a unique value for exactly one case in each class and missing on all other cases yields the largest decrease in impurity (Wei-Yin, 2001). Also, when using linear combination splits, it is impractical to find at every node the best univariate surrogate linear combination split for each possible superset of those in the original split (CART computes only univariate splits regardless of the split option).

The idea of surrogate splitting is reasonable if high correlations among the predictor variables exist. Since the "problem" attribute (the attribute with missing values) is crucially dependent on the surrogate attribute in terms of a high correlation, when the correlation between the "problem" attribute and the surrogate is low, surrogate splitting becomes very clumsy and unsatisfactory. In other words, the method is highly dependent on the magnitude of the correlation between the original attribute

and its surrogate. Surrogate variable splitting relies on redundant attribute variables. Also, it is usually safer for predicting new things to be somewhere between two known possibilities. Surrogate splitting, however chooses one from the two.

3.4.2.2. Fractioning of Cases

Quinlan (1993) borrows the probabilistic complex approach by Cestnik *et al.*, (1987) by "fractioning" instances or cases (FC) based on a *priori* probability of each value determined from the instances at that node that have specified values. Quinlan starts by penalising the information gain measure by the proportion of unknown instances and then splits these instances to both subnodes as follows:

Suppose that T is the total number of cases at a particular node, and T_{miss} is the number of cases with unknown values of attribute A. Let $f = \{T - T_{miss}\}/T$. The definition of gain can now be defined as: $Gain(A) = f \times \{Entropy(T) - Entropy(A)\}$ where Entropy (T) and Entropy (A) are calculated as before (see Section 2.8.1.1) but only instances with known values of *A* are taken into account. Similarly, let *A* split the non-missing cases T into subsets $T_1,..., T_n$. Define the information value of the attribute A as:

$$IV(A) = -\sum_{j=1}^{J_k} \frac{|T_j|}{|T|} \log_2 \frac{|T_j|}{|T|} - \frac{|T_{miss}|}{|T|} \log_2 \frac{|T_{miss}|}{|T|} \qquad (3.13)$$

The gain ratio is again defined as: $GR(A) = \dfrac{Gain(A)}{IV(A)}$.

The learning phase requires that the relative frequencies *f* above from the training set be observed. Each instance *x* of class *C* with an unknown attribute value *A* is substituted. The next step is to distribute the unknown examples according to the proportion of occurrences in the known examples; treating an incomplete observation as if it falls down all subsequent nodes. For example, if an internal node *t* has ten known examples (six examples with t_L and four with t_R), then we would say the probability of $t_L = 0.6$, and the probability of t_R is 0.4. Hence, a fraction of

0.6 of instance x is distributed down the branch for t_L and a fraction 0.4 of instance x to t_R. This is carried out throughout the tree construction process. The evaluation measure is weighted with the fraction of known values to take into account that the information gained from that attribute will not always be available (but only in those cases where the attribute value is known). During training, instance counts used to calculate the evaluation heuristic include the fractional counts of instances with missing values. Instances with multiple missing values can be fractioned multiple times into numerous smaller and smaller "portions".

For classification, Quinlan (1993)'s technique is to explore all branches below the node in question and then take into account that some branches are more probable than others. Quinlan further borrows Cestnik *et al.*'s strategy of summing the weights of the instance fragments classified in different ways at the leaf nodes of the tree and then choosing the class with the highest probability or the most probable classification. Basically, when a test attribute has been selected, the cases with known values are divided on the branches corresponding to these values. The cases with missing values are, in a way, passed down all branches, but with a weight that corresponds to the relative frequency of the value assigned to a branch. Both strategies for handling missing attribute values are used for the C4.5 system.

Despite its strengths, the fractional cases technique can be quite a slow, computationally intensive process because several branches must do the calculation simultaneously. So, if K branches do the calculation, then the central processing unit (CPU) time spent is K times the individual branch calculation.

3.4.2.3 Dynamic Path Generation

Dynamic path generation (DPG) also known as lazy decision tree learning are embedded methods that have been used for handling missing data. White (1987) first described DPG as a method of handling missing values on the testing phase and was further explored by Liu *et al.* (1997). The same principle is also known as Lazy Evaluation, as subsequently described by Friedman *et al.* (1996). Basically, this technique differs from the traditional approach such as C4.5 and CART of inducing a tree from training data and then, as a separate step, applying it to the test data.

That is, only those variables that are non-missing for the test case under consideration are considered for branching. Essentially, the two steps are collapsed to a single operation, although the distinction between the training and test sets is always maintained. With DPG, an actual tree is not built. What is done is to operate on a database of training cases by generating just the path (or rule) necessary to classify the single test case under current consideration. During classification, the algorithm chooses the most informative attribute on which to branch. Any attribute with a missing value is never branched on. Instead, the algorithm tries with the second most informative attribute; if that attribute has a missing value as well it tries the third most informative attribute; and so on. Note that this case-by-case approach makes the technique particularly appropriate for use in conjunction with n-fold (leaving-one-out) cross-validation but it can be computationally demanding. Since in lazy learning you do not train a model until you know the test case, the missingness in test case may 'shadow' values in the training set.

3.4.3 Other Methods

Friedman (1977), and later Quinlan (1989), follow the LD approach when handling incomplete data using DTs. When using this approach, all the instances (cases) with unknown attribute values are omitted from the analysis while forming the split, and the tree is constructed from only those instances which have known attribute values, i.e. complete observation vectors. Evaluating a split using known information improves credibility. Excluding instances with missing values is feasible with univariate trees (trees that use splits based on a single attribute at each internal node) because only observations missing on a single input are excluded at any one time. This compares favourably with other modelling techniques that exclude observations missing any input value, which may leave very few usable observations. Another probabilistic approach to handle missing attribute values is called event-covering. Details about the method can be found in Chiu and Wong (1986) and further described by Wong and Chiu (1987). Genetic programming (Koza, 1993) and thus Strongly Typed Generic Programming (STGP) has been used for learning from examples with incomplete or missing data. Backer (1996)'s approach is learning substituting computations based on input data for the missing values

using STGP. This is motivated by the thought that during the evolutionary process, programs with appropriate substituting computations will have a higher fitness due to their better behaviour on cases with missing values. The correlations between the input data are also utilized by this approach. The resulting program can also work on new cases with missing values. Incremental imputation through tree-based methods is another method that has been used to deal with data presenting missing values in many covariates (Conversano *et al.*, 2002). This approach relies on the assumption that the data is MCAR and uses lexicographic order to rank missing values that occur in different variables and deal with them incrementally, i.e., augmenting the data by the previously filled in instances according to the defined order. This approach overcomes the shortcomings of conditional mean imputation. However, it struggles when data is linear.

Chapter 4

Experiments with Current Methods

4.1 Introduction

One of the major issues in DT learning is classification accuracy on novel instances. Handling missing (incomplete) attribute values is an important issue for decision tree learning, since missing values in either training or test data affect classification accuracy. In fact, increases in missing data proportions would be expected to result in increases in predictive error. In studies where missing data are present, it is common for researchers to use ad hoc approaches such as LD or imputation to deal with the missing data problem. Yet, on the one hand, deleting instances missing values can drastically reduce the sample size, resulting in a severe lack of statistical power and biased results. On the other hand, selecting the appropriate imputation technique can also be a problematic and hard task. Caution should be applied when using imputation-based approaches since these techniques require certain assumptions about the underlying missing data mechanism (i.e. the probability law generating the missing data) to be satisfied. Recently, missing data techniques which utilise machine learning algorithms have been shown to outperform simple statistical methods like mean or mode imputation algorithms, which are methods broadly used to treat missing values. Furthermore, different techniques may work well under different situations.

This chapter is about a substantial comparative simulation study of the effect of different MDTs on the predictive accuracy of the resulting DTs. However, before embarking on that, related studies are briefly reviewed.

4.2 Related Work

Although the problem of incomplete data has been treated adequately in various real world datasets, there are rather few published works or empirical studies concerning the task of learning DTs from incomplete data. From the results of these studies, no missing data technique has been found to be uniformly superior to the others.

Randomness of missing data has been shown to greatly influence the accuracy of missing data techniques (Little and Rubin, 1987; Roth, 1994; Graham and Donaldson, 1993). For example, MCAR data has been shown to be easier to deal with compared with MAR and IM data. In fact, IM data has been proven to be the most difficult to deal with (Little and Rubin, 1987). Other factors influencing missing data techniques accuracy include the pattern and the proportion of missing data. To our knowledge, no single study has considered all of these factors combined. Also, to our knowledge no comprehensive study has been conducted that empirically evaluated the multiple imputation approach for handling incomplete data using decision trees.

An experiment to compare FC, DTSI and MMSI was carried out by Quinlan (1985). This comparison, according to Quinlan, gave unconvincing results even though it did show how the performance of the methods was much worse when several values of several attributes were missing. The DTSI method (which is closest to EMSI since it uses all the attributes in the estimation process) performed better than both the FC and MMSI methods. However, the difference in accuracy between the two methods was quite small. Quinlan (1989) later compared LD, MMSI, FC and DTSI using several datasets. FC outperformed all the other methods so Quinlan decided to use it as a strategy for handling missing values for the implementation of C4.5 (Quinlan, 1993).

Shavlik *et al.* (1991) compared the performance of backpropagation, perceptron and ID3 in case of instances with missing values using four datasets. Their results showed increases in proportion of missing values being associated with decreases in predictive accuracy. Furthermore, backpropagation was shown to be able to handle noise and missing feature values better than ID3 and perceptron do especially for the dataset that had training examples containing only numeric values.

Bruha and Franek (1996) compared experimentally five methods for handling incomplete data. These are: the LD, considering "missing" as an additional regular value, MMSI, FC, and any value of the known attribute values that occur in the training set. Of the five methods, the most successful were the imputation methods, namely, MMSI and any value. This contradicts to some extent Quinlan (1989) which

indicated FC to be the one of the best techniques for handling missing values in ID3. The worst method was found to be LD followed by MMSI.

Lobo and Numao (1999; 2000) evaluated the accuracy performance of DTSI (with ordered attributes), MMSI and FC. For incomplete training and test data, DTSI outperformed both methods with MMSI giving the worst performance. For incomplete test data, no method performed better than the other.

Lakshminarayan *et al.* (1999) performed a simulation study comparing different missing data techniques on industrial databases. Techniques considered were machine learning methods for missing data imputation. The results showed that for the single imputation task, the supervised machine learning algorithm, C4.5 (Quinlan, 1993) which uses the FC procedure performed better than the unsupervised learning algorithm, Autoclass (Cheeseman *et al.*, 1988) while for the multiple imputation task, both methods performed comparably. Also, both methods handled mixed data naturally.

Feelders (1999) experimentally compared the use of imputation (both EMSI and EMMI) and SVS as methods for handling missing data using decision trees in data mining. His results showed both EMMI and EMSI having a superior performance than SVS in terms of predictive accuracy of the resulting models. However, the differences in error rates between EMSI and SVS were found to be very small. Overall, EMMI yielded the best results.

Strike *et al.* (2000) performed a comprehensive simulation study to evaluate three missing data techniques in the context of software prediction. These techniques are LD, MMSI and eight different types of hot-deck single imputation (HDSI). Three missing data mechanisms (MCAR, MAR and IM) were evaluated and two patterns of missing data (univariate and monotone) were simulated. Their results showed LD as not only having a severe impact on regression estimates but yielding a small bias as well. However, the precision of LD worsened with increases in proportion of missingness. Their results further showed that better performance would be obtained from applying imputation techniques. The best performance was obtained by using HDSI with Euclidean distance and a z-score standardization.

Nine different approaches to missing attribute values were compared by Grzymala and Hu (2000). These methods were LD, FC, MMSI, CCSI, a method of assigning all possible values of the attribute, method of assigning all possible values of the attribute restricted to the given class variable or concept, an event covering method, a method of treating missing attribute values as special values and a special LEM2 algorithm (Grzymala and Wang, 1997). Grzymala and Hu (2000) concluded that FC and LD were the best methods among all nine approaches while MMSI achieved the worst performance. In addition, FC outperformed LD on 60 percent of the ten datasets that were used. Otherwise, the remaining methods did not differ significantly from one another. The method of assigning to the missing attribute value all possible values of the attribute and the method of assigning to the missing attribute value all possible values restricted to the same class were found to be excellent approaches. However, the authors argued that they did not have enough evidence to support the claim that these approaches were superior.

Kalousis and Hilario (2000) evaluated seven classification algorithms with respect to missing values: two rule inducers (C5.0-rules and Ripper), one nearest neighbour method, one orthogonal (C5.0-tree), one oblique decision tree algorithm, a naïve Bayes algorithm and a linear discriminant. Various patterns and mechanisms of missingness (MCAR and MAR) in current complete datasets were simulated. This was an excellent idea since the pattern and mechanism of missing values was an important dimension in their study. Their results indicate that naïve Bayes (NB) is most resilient to missing values while the k-nearest neighbour single imputation (*k*NNSI) and FC are more sensitive to missing values. Their results further show that for a given proportion of missing values, the distribution of missing values among attributes is at least as important as the mechanism of missingness.

Another comparative study of missing data techniques in the context of software prediction was carried out by Myrtveit *et al.* (2001). The four missing data techniques were LD, mean imputation (MEI), similar response pattern imputation (SRPI) and FIML. Their results showed FIML performing well for MCAR data. Also, LD, MEI and SRPI were shown to yield biased results for other missing data mechanisms other than MCAR. Their recommendations were to use FIML if one had enough data and to use MEI or SRPI if one needed more data. A combination of LD

with a regression models was recommended for small datasets where FIML cannot be used. In addition, they argued that LD should only be used for MCAR data.

Fujikawa and Ho (2002) evaluated theoretically several methods of dealing with missing values. The methods evaluated were MMSI, linear regression, standard deviation method, *k*NNSI, DTSI, auto-associative neural network, LD, lazy decision tree, FC and SVS. *k*NNSI and DTSI showed good results. In terms of computation cost, MMSI and FC were found to be reasonably good.

Batista and Monard (2003) investigated the effects of four methods of handling missing data at different proportions of missing values. There methods investigated were *k*NNSI, MMSI, and internal algorithms used by FC and CN2 to treat missing data. Missing values were artificially simulated in different rates and attributes into the datasets. *k*NNSI imputation showed a superior performance compared with MMSI when missing values were in one attribute. However, both methods compared favourably when missing values were in more than one attribute. Otherwise, FC achieved a performance as good as *k*NNSI.

The performance of *k*NNSI and MMSI was analysed by Cartwright *et al.* (2003) using two small industrial datasets. Their results showed both methods yielding good results with *k*NNSI providing a more robust and sensitive method for missing value estimation than MMSI.

Farhangfar *et al.* (2004) compared five imputation methods; two of the methods are based on statistical algorithms (MMSI and HDSI) while the remaining three are based on machine learning algorithms (rule based, NB, FC). Missing data was artificially generated in all attributes (including the class attribute) to model only the MCAR mechanism. This was done for different proportions of missing data. FC achieved the best overall performance, closely followed by NB. MMSI was found to be stable, i.e. its performance degradation was the slowest compared to the other methods. Their results further showed that the performance of methods like MMSI and HDSI does not depend on the number of attributes, which conforms to the procedure they use. MMSI was also found to be the fastest in terms of computational time, followed by FC.

Song and Sheppered (2004) evaluated *k*NNSI imputation and CCSI for different patterns and mechanisms of missing data. Their resuls showed *k*NNSI slightly outperforming CCSI with the missing data mechanisms having no impact on either of the two imputation methods.

Sentas *et al.* (2004) proposed using multinomial logistic regression imputation (MLRI) as a new technique for handling missing categorical values. Their proposed procedure was compared with LD, MMSI, EMSI and regression-based single imputation (RBSI). Their results showed LD and MMSI as efficient when the percentage of missing values is small while RBSI and MLRI outperformed both LD and MEI as the amount of missing values increased. Overall, MLRI gave the best results, especially for MCAR and IM data. For MAR data, MLRI compared favourably with RBSI.

In conclusion, we think that the available prior research supports a lot of the questions we are targeting with our study. First, there are no substantial differences in accuracy among MDTs at lower levels of missing data. However, the performance of each MDT declines when the percentage of missing values increases. While ad hoc methods such as LD and MMSI are easy to use and are more appropriate for MCAR data, they seem to lack the accuracy for dealing with missing data more generally. Model-based approaches such as FIML and the EM algorithm seem to be generally superior to ad hoc methods in that they are statistically efficient. In addition, model-based approaches to missing data estimation are expected to be much more powerful and reliable than ad hoc methods because they utilize all the information and relationships with the data matrix.

According to the above studies, among imputation techniques, the results are not so clear. However, machine learning methods appear to achieve higher accuracy than traditional statistical approaches because of their complicated processing. However, they take much more time in processing than statistical methods do. In addition, statistical methods appear to be more stable with respect to increasing amount of missing data, i.e., they show less deterioration in performance with increasing amount of missing data compared to machine learning methods. Also, multiple

imputation, which overcomes limitations of single imputation seem not to have been widely adopted by researchers even though it has been shown to be flexible and software for creating multiple imputations is available. Finally, results from previous studies suggest that results achieved using simulated data are very sensitive to the MAR assumption. Hence, if there is a reason to believe that if the MAR assumption does not hold, alternative methods should be used.

There are several very interesting ramifications of previous studies. Firstly, some studies found LD (the default in many statistical programs) as equally accurate and sometimes more efficient than machine learning methods such as FC and NB. However, LD carried the penalty of a larger loss in statistical power (the ability of a statistical test to detect a pattern in a dataset). The accuracy of LD probably stems from the fact that deletion techniques result in data matrices that mirror the true data structure. The superior performance of *k*NNSI to methods like FC and NB on small datasets is rather surprising. The *k*NNSI procedure should not be as effective due to the fact that, for small data sets, one runs the risk of using the same donor many times, thus resulting in a loss of precision in the imputed value. Even more interesting is the good performance of RBSI for MAR data even though it is more appropriate for MCAR data.

4.3 General Experimental Set-Up

One of the objectives of this thesis is to investigate the robustness and accuracy of methods for tolerating incomplete data using tree-based models. This section describes experiments that were carried out in order to compare the performance of the different approaches previously proposed for handling missing values in both the training set and test (unseen) set. The effects of different proportions of missing values when building the tree (training) and when classifying new instances (testing) are further examined, experimentally. Finally, the impact of the nature of different missing data mechanisms on the classification accuracy of resulting trees is examined. A combination of small and large datasets, with a mixture of both nominal and numerical attribute variables, was used for these tasks. All datasets

have no missing values. The main reason for using datasets with no missing values is to have total control over the missing data in each dataset.

The simulation study concentrates on performing experimental analysis of MDTs, which range from simple statistical algorithms to machine learning algorithms. These techniques are divided into the following categories: Ignoring and discarding data (LD), imputation (CCSI, DTSI, EMSI, MMSI, EMMI) and machine learning (FC, SVS). All eight MDTs were used as training methods, i.e., methods for building trees from incomplete data with only seven approaches (LD, DTSI, EMSI, MMSI, EMMI, SVS and FC) used as test methods, i.e., methods for classifying incomplete vectors using DTs. Table 4.1 summarizes the MDTs to be investigated and the section where each technique is discussed in the thesis.

Table 4.1 Missing data techniques to be investigated

Technique	Acronym	Section
Discarding or ignoring data:		
Listwise deletion	LD	3.3.1.1
Single imputation:		
Expectation Maximization	EMSI	3.3.2.1.4
Mean or Mode	MMSI	3.4.1.1.1
Conditioning on Class	CCSI	3.4.1.1.2
Decision Tree*	DTSI	3.4.1.1.6
Multiple imputation:		
Expectation Maximization	EMMI	3.4.1.2
Machine learning:		
Surrogate Variable Splitting	SVS	3.4.2.1
Fractioning of Cases	FC	3.4.2.2

* It can also be considered a machine learning technique

The class variable is always available in the training set but not in the test set. Two of the above-mentioned methods rely on the class variable as a technique of handling missing values. The CCSI method is one approach that relies heavily on the conditioning of class and was considered as a method of handling missing values when they are only in the training set. DTSI is another procedure that relies on the class variable to estimate the missing value of a particular attribute using a decision tree. For the test set, the "class" problem was dealt with by using all the available attributes (and not the class variable) to estimate the missing values.

To perform the experiments each dataset was split randomly into five mutually exclusive parts (Part I, Part II, Part III, Part IV, and Part V) of equal (or approximately equal) size. 5-fold cross validation was used for the experiment. For each fold, four of the parts of the instances in each category were placed in the training set, and the remaining one was placed in the corresponding test set as shown in Table 4.2. The same splits of the data were used for all the methods for handling incomplete data.

Since the distribution of missing values among attributes and the missing data mechanism were two of the most important dimensions of this study, three suites of data were created, corresponding to MCAR, MAR and IM. In order to simulate missing values on attributes, the original data bases are run using a random generator (for MCAR) and an attribute pairs and percentile approach (for both MAR and IM, respectively). See below: both of these procedures have the same percentage of missing values as their parameters. The random generator and the attribute pairs and percentile procedures were run to get datasets with four levels of proportion of missingness p, i.e., 0%, 15%, 30% and 50% missing values.

To maintain a level of consistency with the proportion of missing values simulated, only datasets whose percentage missings came out to be close to the nominal percentage missing were simulated. Otherwise, those that were not close were rejected and not considered in the analysis. To carry out this task, some form of truncated binomial distribution was used, i.e., any percentage value that was outside the original binomial and truncated binomial distribution regions was

rejected. Any value less than or greater than 0.5% to the specific level of missingness being looked at was not considered in our analysis.

Table 4.2 Partitioning of dataset to training and test sets

	Training Set	*Test Set*
Fold 1	Part II + Part III + Part IV + Part V	Part I
Fold 2	Part I + Part III + Part IV + Part V	Part II
Fold 3	Part I + Part II + Part IV + Part V	Part III
Fold 4	Part I + Part II + Part III + Part V	Part IV
Fold 5	Part I + Part II + Part III + Part IV	Part V

Each of these missingness proportions was obtained using each of the different missing value mechanisms (MCAR, MAR and IM). The experiment consists of having three different combinations: having *p%* of data missing from both the training and test sets; having *p%* of data missing from the training set and a complete test set; and having *p%* missing in the test set and a complete training set. This was carried out for each dataset and 5-fold cross validation was used.

The missing data mechanisms were constructed by generating a missing value template (1= present, 0 = missing) for each attribute and multiplying that attribute by a missing value template vector. Our assumption is that the observations or instances are independent selections.

For each dataset, two suites were created. First, missing values were simulated on only one attribute variable (univariate pattern); this was the attribute that was most highly correlated with the class variable. Second, missing values were introduced on all the attribute variables (arbitrary pattern with missingness was evenly and uniformly distributed across all the attributes). This was the case for the three missing data mechanisms, which from now shall be called MCAR*univa*,

MAR*univa*, IM*univa* (for the former), on the one hand, and MCAR*unifo*, MAR*unifo*, IM*unifo* (for the latter), on the other hand.

These procedures are described as follows:

MCAR

Each vector in the template (values of 1's for non-missing and 0's for missing) was generated using a random number generator, utilising the Bernoulli distribution. The missing value template is then multiplied by the attribute of interest, thereby causing missing values to appear as zeros in the modified data.

MAR

Simulating MAR values was more difficult. As mentioned in the previous chapter, the mechanism of MAR values depends only on other observed data and not at all on unobserved or potential data, including the missing data themselves. So, since the idea is to condition the generation of missing values based upon the distribution of the observed values. Attributes of a dataset are separated into pairs, say, (A_X, A_Y), where A_Y is the attribute into which missing values are introduced and A_X is the attribute on the distribution of which the missing values of A_Y is conditioned, i.e., $P(A_Y = \text{miss} \mid A_X = \text{observed})$. Therefore, the first half of attributes would not have missing values and on the second half the missing values of a given attribute would be imputed based on the values of a specific attribute from the first half. Since we wanted to keep the percentage of missing values at the same level overall, we had to alter the percentage of missing values of the individual attributes. Thus, in the case of $k\%$ of missing values over the whole dataset, 2k% of missing values were simulated on A_Y. For each of the A_X attributes its $4k$ percentile was estimated. Then all the instances were examined and whenever A_X attribute has a value lower than the $4k$ percentile a missing value on A_Y is imputed with probability 0, and 1 otherwise. More formally, $P(A_Y = \text{miss} \mid A_X < 4k) = 0$ or $P(A_Y = \text{miss} \mid A_X > 4k) = 1$. This technique generates a missing value template which is then multiplied with A_Y. Once again, the attribute chosen to have missing values was the one most

highly correlated with the class variable. Here, the same levels of missing values are kept.

IM

In contrast to the MAR situation outlined above where data missingness is explainable by other measured variables in a study, IM data arise due to the data missingness mechanism being explainable, and only explainable by the very variable(s) on which the data are missing. For conditions with data IM, a procedure identical to MAR was implemented. However, for the former, the missing values template was created using the same attribute variable for which values are deleted in different proportions.

For consistency, missing values were generated on the same variables for each of the three missing data mechanisms. This was done for each dataset.

Methods used for handling missing values that had been generated using the above-mentioned three missing data mechanisms shall now be looked at.

The LD, SVS and FC procedures are the only three embedded methods that do not estimate the missing value or are not based on "filling in" a value for each missing datum when handling either incomplete training and test data or either of the two. These three methods have already been discussed in Chapter 3. However, programs and code that were used for the methods are briefly described below.

No software or code was used for LD. Instead, all instances with missing values on that particular attribute were manually excluded or dropped, and the analysis was applied only to the complete instances.

For the SVS method, a recursive partitioning (RPART) routine, which implements within S-PLUS many of the ideas found in the CART book and programs of Breiman *et al.* (1984) was used for both training and testing decision trees. This programme, which handles both incomplete training and test data, is by Therneau and Atkinson (1997).

The decision tree learner C4.5 was used as a representative of the FC or probabilistic technique for handling missing attribute values in both the training and test samples. This technique is probabilistic in the sense that it constructs a model of the missing values, which depends only on the prior distribution of the attribute values for each attribute tested in a node of the tree. The main idea behind the technique is to assign probability distributions at each node of the tree. These probabilities are estimated based on the observed frequencies of the attribute values among the training instances at that particular node.

The remaining five methods are pre-replacing methods, which use estimation as a technique of handling missing values, i.e., the process of "filling in" missing values in instances using some estimation procedure.

The DTSI method uses a decision tree for estimating the missing values of an attribute and then uses the data with filled values to construct a decision tree for estimating or filling in the missing values of other attributes. This method makes sense when building a decision tree with incomplete data, the class variable (which plays a major role in the estimation process) is always present. For classification purposes (where the class variable is not present), first, imputation for one attribute (the attribute highly correlated with class) was done using the mean (for numerical attributes) or mode (for categorical attributes), and then the attribute was used to impute missing values of the other attributes using the decision tree single imputation technique. In other words, two single imputation techniques were used to handle incomplete test data. An S-PLUS code that was used to estimate missing attribute values using a decision tree for both incomplete training and test data was developed.

The CCSI method conditions the particular attribute with missing value(s) on class. Again, some very simple S-PLUS code for this method was developed. The method fills the missing values with the mean or mode, depending on the type of attribute with the missing values, i.e. whether the attribute is nominal (whereby the class-conditional mode is used) or continuous (whereby the class-conditional mean is used). Notice that this method replaces each missing value with a single plausible

value (more like single imputation). This method can be used only for building trees given incomplete data, especially in real world problems where classes to which the instances to be classified belong are not known. The reader is referred to Section 3.5.1.1.2 for more details about this method.

S-PLUS code was also developed for the MMSI approach. The code was developed in such a way that it replaced the missing data for a given attribute by the mean (for numerical or quantitative attribute) or mode (for nominal or qualitative attribute) of all known values of that attribute.

There are many implementations of MI. Schafer's (1997) set of algorithms (headed by the NORM program) that use iterative Bayesian simulation to generate imputations was an excellent option. NORM was used for datasets with only continuous attributes. A program called MIX written is used for mixed categorical and continuous data. MIX is an extension of the well-known general location model. It combines a log-linear model for the categorical variables with a multivariate normal regression for the continuous ones. For strictly categorical data, CAT was used. All three programs are available as S-PLUS routines. Schafer (1997), and Schafer and Olsen (1998) gives details of the general location model and other models that could be used for imputation tasks.

Due to the limit of the dynamic memory in S-PLUS for Windows (S-PLUS, 2003) when using the EM approach, all the big datasets were partitioned into subsets, and S-PLUS run on one subset at a time. Our partitioning strategy was to put variables with high correlations with close scales (for continuous attributes) into the same subset. This strategy made the convergence criteria in the iterative methods easier to set up and very likely to produce more accurate results. The number of attributes in each subset depended on the number of instances and the number of free parameters to be estimated in the model, which included cell probabilities, cell means and variance-covariances. The number of attributes in each subset was determined in such a way that the size of the data matrix and the dynamic memory requirement was under the S-PLUS limitation and the number of instances was large relative to the number of free parameters. Separate results from each subset

were then averaged to produce an approximate EM-based method which are substituted for (and continue to call) EM in our investigation.

To measure the performance of methods, the training set/test set methodology is employed. For each run, each dataset is split randomly into 80% training and 20% testing, with different percentages of missing data (0%, 15%, 30%, and 50%) in the covariates for both the training and testing sets.

Even though our experiment covers a wide range of separate and completely crossed factors, the effects of certain specific combinations are looked at. Firstly, the situation of having both incomplete training and test data and the effect this has on resulting trees was looked at. Secondly, the situation of having incomplete training data but complete test data was investigated. Lastly, the effect of having only incomplete test data was considered. The combination of having incomplete training data and complete test data was still considered even though one would expect test data to be incomplete when training data are incomplete. In fact, it would be a rare situation where the training data were incomplete but the test data were complete (Hand, 2000).

A classifier was built on the training data and the predicted accuracy is measured by the smoothed error rate of the tree, and was estimated on the test data. This procedure was repeated 5 times as described below. The reader is referred to Section 2.8.4 for the definition of smoothed error rate and the important reasons why such an error rate was used for the experiments.

Trees on complete training data were grown using the *Tree* function in S-PLUS (Becker *et al.*, 1988, Venables and Ripley, 1994). The function uses the *GINI* index of impurity (Breiman *et al.*, 1984) as a splitting rule and cross validation cost-complexity pruning as pruning rule. Accuracy of the tree, in the form of a smoothed error rate, was predicted using the test data.

For incomplete training data the eight training methods for handling incomplete data, already described in previous sections, were used. Seven methods were used for handling incomplete test data. For split selection, the impurity approach was

used. For quantitative (ordered) predictors the approach searches over all possible values c for splits of the form $\{X \leq c\}$ or $\{X > c\}$to define the left and right child nodes, where c is a real number ranging over (-∞, ∞). The training instances in the partition based on the values of the attribute being considered for splitting are then sorted. Let $c_1,...,c_n$ be the sorted values of a numeric attribute A. Investigate the midpoint of each interval c_i to c_{i+1} as a possible split point. For qualitative or categorical predictors with attribute X taking values in $\{b_1,...,b_L\}$, the search is over all splits of the form $\{X \in c\}$ or $\{X \notin c\}$ where c is a non-empty subset of $\{b_1,...,b_L\}$. For detailed information about splits for numeric and categorical attributes, individually, the reader is referred to Sub-Section 2.2.1.

For pruning, a combination of 10-fold cross validation cost complexity pruning and 1 Standard Error (1-SE) rule (Section 2.3.3) to determine the optimal value for the complexity parameter was used. The same splitting and pruning rules when growing the tree were carried out for each of the twenty one datasets.

It was reasoned that the condition with no missing data should be used as a baseline and what should be analysed is not the error rate itself but the increase or excess error induced by the combination of conditions under consideration. Therefore, for each combination of method for handling incomplete data, the number of attributes with missing values, proportion of missing values, and the error rate for all data present was subtracted from each of the three different proportions of missingness. This would be the justification for the use of differences in error rates analysed in some of the experimental results.

Another point to note is the reason for using difference in error rates when making comparisons between MDTs instead of, say, division or ratios of error rates. First, differences are natural and understandable scale in this context, that is, people would understand a "p percentage point" worsening in error rate to mean a simple addition of p%. Secondly, ratios of error rates would lead to statements like "A increases error rate by p%" which would be misinterpreted as meaning a p%

difference in error rate. Finally, the ANOVA assumes the error rates to be on an additive rather than multiplicative scale.

All statistical tests were conducted using the MINITAB statistical software program (MINITAB, 2002). Analyses of variance, using the general linear model (GLM) procedure (Kirk, 1982) were used to examine the main effects and their respective interactions. This was done using a 4-way repeated measures designs (where each effect was tested against its interaction with datasets). The fixed effect factors were the: missing data techniques; number of attributes with missing values (missing data patterns); missing data proportions; and missing data mechanisms. A 1% level of significance was used because of the many number of effects. The twenty one datasets used were used to estimate the smoothed error. Results were averaged across five folds of the cross-validation process before carrying out the statistical analysis. The averaging was done as a reduction in error variance benefit.

A summary of all the main effects and their respective interactions are provided in the Appendix in the form of Analysis of Variance (ANOVA) tables.

4.3.1 Datasets

This section describes the twenty one datasets that were used in the experiments to explore the impact of missing values on the classification accuracy of resulting decision trees. All twenty one datasets were obtained from the Machine Learning Repository maintained by the Department of Information and Computer Science at the University of California at Irvine (Merz *et al.*, 1996). They are summarized in Table 4.3. The first eight involve datasets with only two classes and the last thirteen involve datasets with more than two classes.

As shown in Table 4.3, the selected twenty one datasets cover a comprehensive range for each of the following characteristics:

- the size of datasets, expressed in terms of the number of instances ranges between 57 and 20000
- the number of attributes ranges between 4 and 60

- the number of classes ranges between 2 and 26
- the number of the type of attributes (numerical or nominal or both)

In general, the datasets were selected in order to assure reasonable comprehensiveness of the results.

Table 4.3 Datasets used for the experiments

Dataset	Instances	Attributes		Classes
		Ordered	*Nominal*	
Two classes:				
German	1000	7	13	2
glass (G2)	163	9	0	2
heart-statlog	270	13	0	2
Ionosphere	351	31	1	2
kr-vs-kp	3196	0	36	2
Labor	57	8	8	2
pima-indians	768	8	0	2
Sonar	208	60	0	2
More than two classes:				
Balance scale	625	4	0	3
Iris	150	4	0	3
waveform	5000	40	0	3
lymphography	148	3	15	4
Vehicle	846	18	0	4
Anneal	898	6	32	5
Glass	214	9	0	6
satimage	6435	36	0	6
Image	2310	19	0	7
Zoo	101	1	15	7
LED 24	1500	0	24	10
Vowel	990	10	3	11
Letter	20000	16	0	26

A very brief description of each dataset is presented below:

anneal

This dataset consists of 898 instances with 6 numeric, 14 binary and 18 nominal attributes. The task is to predict the steel annealing behaviour into 5 classes.

balance scale

This dataset was generated to model psychological experimental results. Each instance is classified as having the balance tip to the right, tip to the left, or be balanced. The attributes are left weight, the left distance, the right weight, and the right distance. The correct way to find the class is the greater of *left_distance* x *left_weight* and *right_distance* x *right_weight*. If they are equal, it is balanced. This is the 3 class problem, with 625 instances and 4 numeric attributes.

german

This data set concerns credit applications. There are 1000 credit applicants (instances) described by 20 attributes, which are a mixture of 7 continuous (for example, age, credit amount, duration of account) and 13 nominal attributes (for example, marital status and sex, job, reason for loan request) and two classes. This data set represents the problem of predicting of a good or bad credit applicant.

glass

The task of this dataset is to identify a glass sample taken from the scene of an accident as one of six types of glass. Each case consists of 9 chemical measurements on one of 6 type of glass. There are 214 observations.

glass (G2)

This data set consists of 163 instances and it represents the problem of determining if a given piece of glass is 'float processed' or 'non-float processed'. The input vector consists of 9 continuous valued attributes where each attribute indicates the concentration of a given element (Mg, Na, A1...) and one attribute indicates the refractive index.

heart-statlog

The purpose of the data set is to predict the presence or absence of heart disease given the results of various medical tests carried out on a patient. There are 2 classes, 13 numeric attributes and 270 instances.

image

The task for this domain is to predict 7 types of outdoor images. The images are hand-segmented to create a classification for every pixel. Each of the 2310 instances is a 3 x 3 region, described by 19 numerical features by 7 classes. The classes are brickface, sky, foliage, cement, window, path and grass. The region-pixel-count feature was found to be constant for all the 2310 instances, henceforth, was removed from the analysis leaving 18 numerical features by 7 classes.

ionosphere

The radar data was collected by a system in Goose Bay, Labrador. This system consists of a phased array of 16 high-frequency antennae with a total transmitted power of the order of 6.4 kilowatts. The targets were free electrons in the ionosphere. 'Good' radar returns are those showing evidence of some type of structure in the ionosphere. 'Bad' returns are those that do not; their signals pass through the ionosphere. The data set has 351 instances described by 33 numerical attributes, 1 binary attribute and 2 classes.

iris

This is a complete data set with petal and sepal width, and petal and sepal length as the attributes. There are 150 instances each described by 4 numeric features and three classes. The classes are iris setosa, iris versicolar and iris virginica. For the Iris data set, the task is to predict the type of iris plant.

kr-vs-kp

The task of this domain is to recognise illegal chess positions in the KRK chess end game. The goal is to learn the concept of in illegal white-to-move position with only white king, white rook and black king on the board. The domain contains a total of 3196 instances, 2 classes, 35 binary attributes and 1 nominal attribute.

labor

The data includes all collective agreements reached in the business and personal services sector for locals with at least 500 members in Canada. The task of this

domain is to make final settlements in labor negotiations in a Canadian industry by using a two-tiered approach with learning from positive and negative instances. There are 2 classes (good or bad), 8 numeric attributes, 3 binary attributes, 5 nominal attributes and 57 instances.

LED24

Breiman *et al.* (1984) artificial data for the digit recognition problem consists of 10 classes representing which of the digits 0-9 is showing on an LED display. The version we are using contains 24 nominal attributes, representing 24 light-emitting diodes. Each of the attributes has a 10% probability of having its value inverted. 1500 cases were randomly generated.

letter

The objective of this dataset is to identify each of a large number of black-and-white rectangular pixel displays as one of the 26 capital letters in the English alphabet. The character images are based on 20 different fonts and each letter within these 20 fonts is randomly distorted to produce a file of 20,000 unique stimuli. Each stimulus is then converted into 16 primitive numerical attributes (statistical moments and edge counts) which are then scaled to fit into a range of integer values from 0 through 15.

lymphography

The dataset was obtained from the University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia. The class of each instance indicates whether lymphography was normal, metastasis, malign lymph or fibrosis. The data consists of 148 records, 3 numerical attributes, 9 binary attributes, 6 nominal attributes and 4 classes.

pima-indians

The Pima Indians data come from the National Institute of Diabetes and Digestive and Kidney Diseases in Maryland, USA. Each of 768 instances is described in 8 numerical features and two classes. The classes are interpreted as testing positive or

negative for diabetes. The task is to decide whether a patient shows signs of diabetes according to World Health Organisation criteria.

satimage

The database consists of the multi-spectral values of pixels in 3x3 neighbourhoods in a satellite image, and the classification associated with the central pixel in each neighbourhood. The aim is to predict this classification, given the multi-spectral values. In the sample database, the class of a pixel is coded as a number. There are 6 classes, 36 numeric attributes and 6435 instances. The data set was used in the Statlog project.

sonar

This is a data set used by Gorman and Sejnowski (1998) in their study of the classification of sonar signals using a neural network. The dataset contains signals obtained from a variety of different aspect angles spanning 90 degrees for metal cylinder and 180 degrees for the rock. The task is to discriminate between sonar signals bounced off a metal cylinder and those bounced off a roughly cylindrical rock. This data set contains 208 instances, 60 continuous attributes and 2 classes.

vehicle

The dataset comes from the Turing Institute, Glasgow, Scotland, UK. The problem is to classify a given silhouette as one of four types of vehicle, using a set of features extracted from the silhouette. The vehicle may be viewed from one of many different angles. The dataset has 4 classes, 18 numeric attributes and 846 observations.

vowel

The task is to recognize the eleven steady-state vowels of British English independent of the speaker. There are 10 numeric attributes describing each vowel. 8 speakers were used to form a training set, and 7 different speakers were used to form an independent test set. Each of the 15 speakers said each vowel six times creating 528 instances in the training set and 462 in the test set. For runs using this

data set we retained the original training and test sets and therefore did not perform a cross validation.

waveform

This is an artificial domain with 3 classes based on 3 waveforms. Each class consists of a random convex combination of two of these waveforms and each instance is generated by added random noise (mean 0, variance 1) in each attribute. It was defined by Breiman *et al.* 1984. This dataset has two versions. The version used for the experiments in the thesis is the one that contains 40 numerical attributes and 5000 records.

zoo

This is an artificial data set from Richard Forsyth described by 7 classes of animals, 1 numerical and 15 binary attributes. The data set has 101 records, each one a different animal. The task is to identify animals at a zoo given their characteristics, such as number of legs, whether the animal can fly, and so on.

4.4 Experimental Results

Experimental results on the effects of current methods for handling both incomplete training and test data on predictive accuracy using DTs are described. The behaviour of these methods is explored for different levels of missing values, and for the MCAR, MAR and IM mechanism of missing data.

The results are presented in three parts. The first part of this section compares the performance of seven different approaches for both building (training) DTs and classifying (testing) incomplete vectors using trees, and further looks at the overall results of each method, averaged for all twenty one datasets. Experimental comparison of training and testing methods on selected individual datasets with purely numerical attributes, purely categorical attributes and mixed attributes, individually, are also presented in this section.

Section 4.2.2 looks at the overall performance of the eight training methods (i.e. when the test set is complete), averaged for all twenty one datasets. Finally, the results of a simulation study looking at the impact of missing values when they occur only in the test set are presented in Section 4.2.3. These overall results are of each testing method, and averaged for all twenty one datasets.

Since the amount of experimental results is very large, to conserve space, only a subset of all results have been reported. Otherwise, other results can be found in the Appendix. Figures plot the classification error of the instances learned on each one of the target domains, averaged over five-fold cross validation runs, by each one of the methods. The same folds were used to evaluate each method.

4.4.1 Overall Results - Incomplete Training and Test Data

Figure 4.1 summarises the error rates of each method against three amounts of missing values. The error rates of each method of the introduced missing values are averaged over the 21 datasets.

For MCAR*univa* data, EMMI has on average the best accuracy while LD exhibits one of the biggest increases in error (Figure 4.1A). From Figure 4.1B, most of the methods achieve slightly bigger error rate increases for MCAR*unifo* data compared with MCAR*univa* data. In the MAR*univa* suite, the behaviour of methods is not very different from the one observed in the MCAR*univa* case (Figure 4.1C). From Figure 4.1D, the performance of methods for MAR*unifo* data is very similar to the one observed for MCAR*unifo* data. Figure 4.1E shows bigger increases in error rates for all the methods for IM*univa* data compared with MCAR*univa* and MAR*univa* data, individually. The performance of all the methods, on average, worsens for IM*unifo* data. Once again, EMMI proves to be the best method at all levels of missing with LD exhibiting the worst performance with an excess error rate of 23.9% at the 50% level (Figure 4.1F).

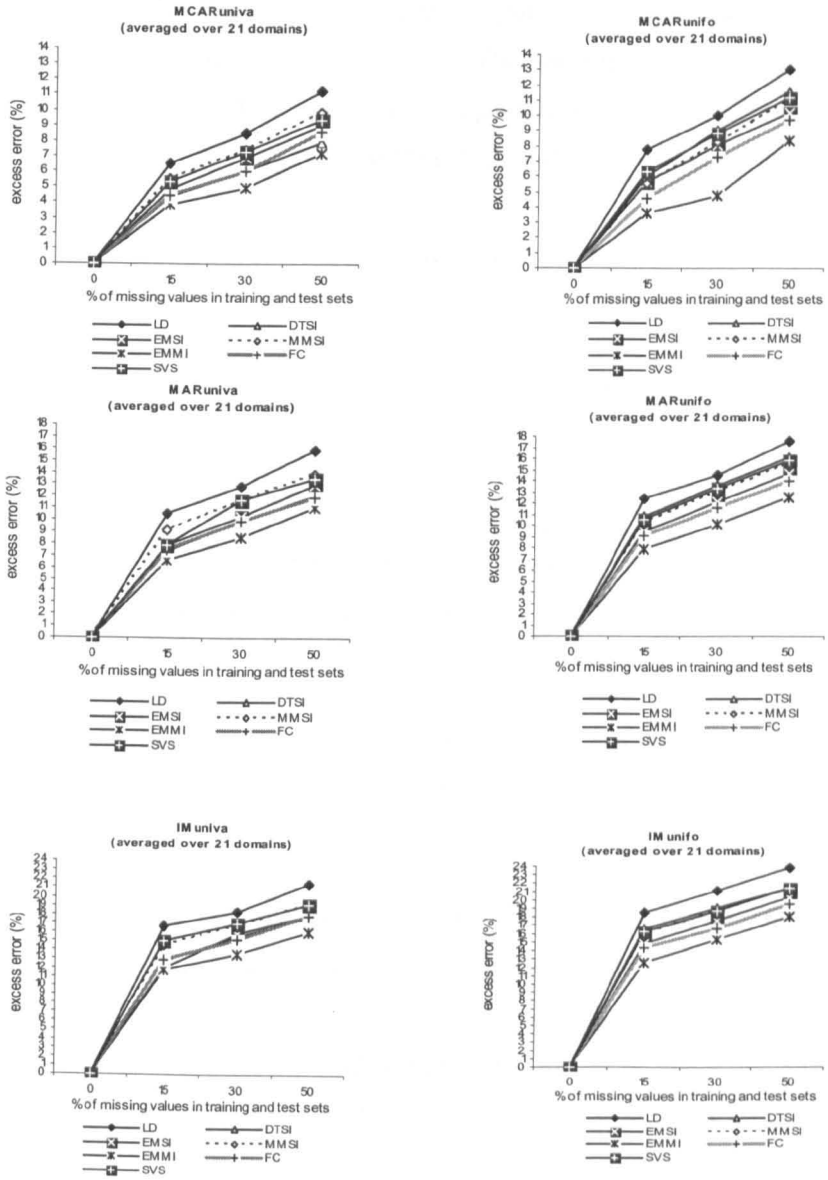

Fig. 4.1. Effects of missing values in training and test data on the excess error for methods over the 21 domains. A) MCAR*univa*, B) MCAR*unifo*, C) MAR*univa*, D) MAR*unifo*, E) IM*univa*, F) IM*unifo*

Main Effects

As shown in Table 4.3 in the Appendix, all the main effects (training and testing methods, number of attributes with missing values, missing data proportions and missing data mechanisms) were found to be significant at the 1% level.

From Figure 4.2, EMMI represents a superior approach to missing data while LD is substantially inferior to the other techniques. The second best method is FC, closely followed by EMSI, DTSI, MMS and SVS, respectively. However, there appears no clear 'winner' between DTSI, EMSI, MMSI, FC and SVS in terms of classification accuracy.

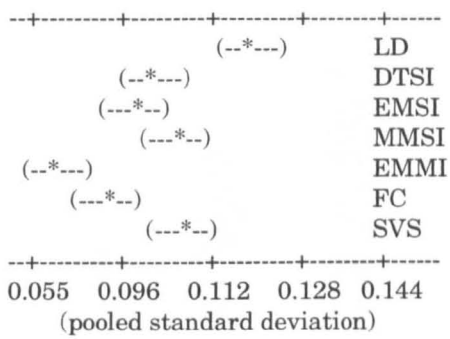

Fig. 4.2. Comparison for training and testing methods: confidence intervals of mean error rates (*)

From Figure 4.3, it appears that missing values have a greater effect when they are distributed among all the attributes compared with when missing values are on a single attribute variable.

The results for proportion of missing values in both the training and test sets show increases in missing data proportions being associated with increases in error rates (Fig. 4.4). In fact, the error rate increase when 50% of values are missing in both the training and test sets is about one and a half times as big as the error rate increase when 15% of values are missing on both sets.

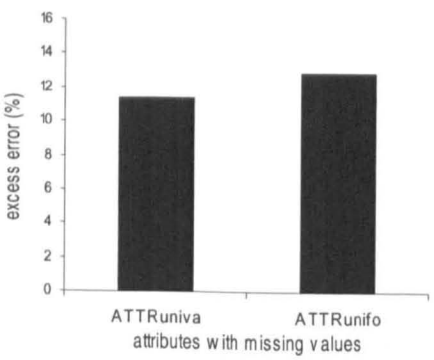

Fig. 4.3. Overall means for number of attributes with missing values

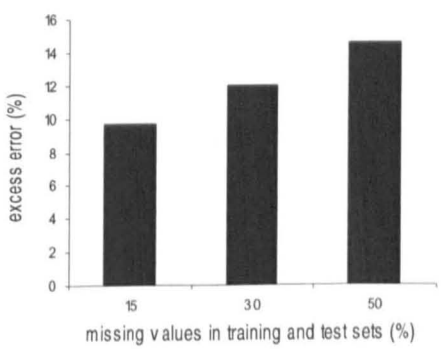

Fig. 4.4. Overall means for missing data proportions

From the results presented in Figure 4.5, IM values entail more serious deterioration in predictive accuracy compared with randomly missing data (i.e. MCAR or MAR data). Overall, MCAR data have a lesser impact on classification accuracy with an error rate increase difference of about 4% (when compared with MAR data) and a much bigger difference of about 10% (when compared with IM data).

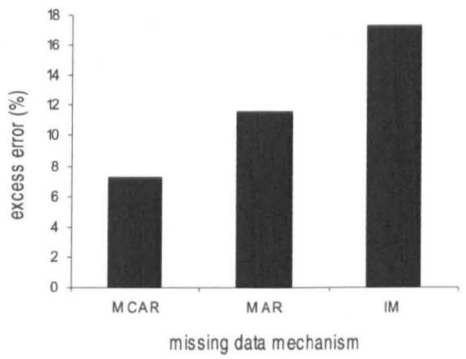

Fig. 4.5. Overall means for missing data mechanisms

Interaction effects

The interaction effect between methods for handling incomplete training and test data and the number of attributes with missing values is displayed in Figure 4.6. From the figure, it follows that all methods perform differently from each other with bigger error rate increases observed when missing values are in all the attributes compared with when they are on a single attribute variable.

A severe impact of having missing values in only one attribute and having missing values in all the attributes is observed for DTSI, LD and SVS, while for the remaining methods the impact is not clear. Once again, the results show EMMI as the best technique for handling incomplete training and test data while LD is the most ineffective method.

From Figure 4.7, the performance by methods do not differ much at lower levels of missing values but vary noticeably as the amount of missing values increases.

As observed earlier, all the methods are more severely impacted by IM data compared with MCAR or MAR data (Figure 4.8). In addition, all the methods are more effective for dealing with MCAR data.

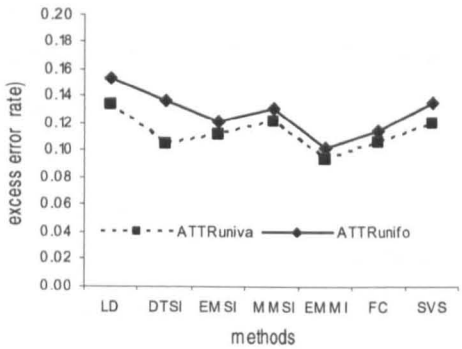

Fig. 4.6. Interaction between methods and number of atributes with missing values

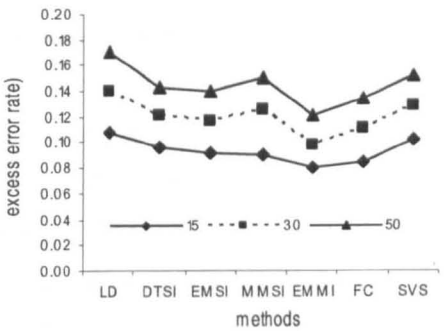

Fig. 4.7. Interaction between methods and proportion of missing values

The results of the interaction between the number of attributes with missing values and missing data proportions show increases in missing data proportions being associated with slightly greater increases in excess error rate (Figure 4.9). Once again, missing values appear to have more impact when they are distributed in all the attributes than when they are in only one attribute.

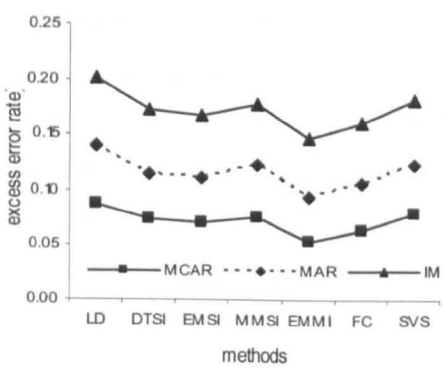

Fig. 4.8 Interaction between methods and missing data mechanisms

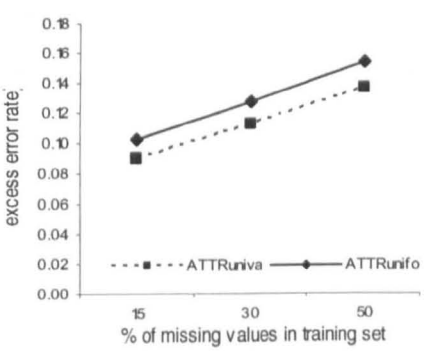

Fig. 4.9. Interaction between number of attributes with missing values and proportion of missing values

4.4.1.1 Results for Individual Datasets - Incomplete Training and Test Data

Based on the experimental results in the previous section, some knowledge was generated about the behaviour of methods for handling incomplete data given different database characteristics. Some methods depended not only on the number of classes but on the number of instances for each class. Some methods were more effective for handling numerical attributes while others achieved good results for nominal attribute or mixed attributes. Some methods gave good results generally, irrespective of the type of attribute. The results that illustrate specific deviations from the overall results of the effectiveness of methods on different database characteristics, mainly numerical, nominal and mixed datasets, are given below. The chosen datasets are typical of their type from each of the three groups of datasets (i.e., numerical, nominal and mixed).

4.4.1.1.1 Results on a dataset with purely numerical attributes: letter

From Figure 4.10A, the effect of the percentage of missing values in MCAR*univa* data is clear. The best overall performance at lower levels of missing values is DTSI while EMMI is most effective at higher levels. EMMI exhibits the smallest error rate increases for MCAR*unifo* (Figure 4.10B). Good performances are observed for EMMI and DTSI while FC struggles with MAR*univa* data (Figure 4.10C). MMSI becomes more effective as the percentage of missing values increases for MAR*unifo* data (Figure 4.10D).The impact of IM*univa* data shows DTSI achieving the best overall performance (Figure 4.10E). From Figure 4.10F, the performance of LD deteriorates with increases in the amount of missing data while FC becomes more robust to missing values.

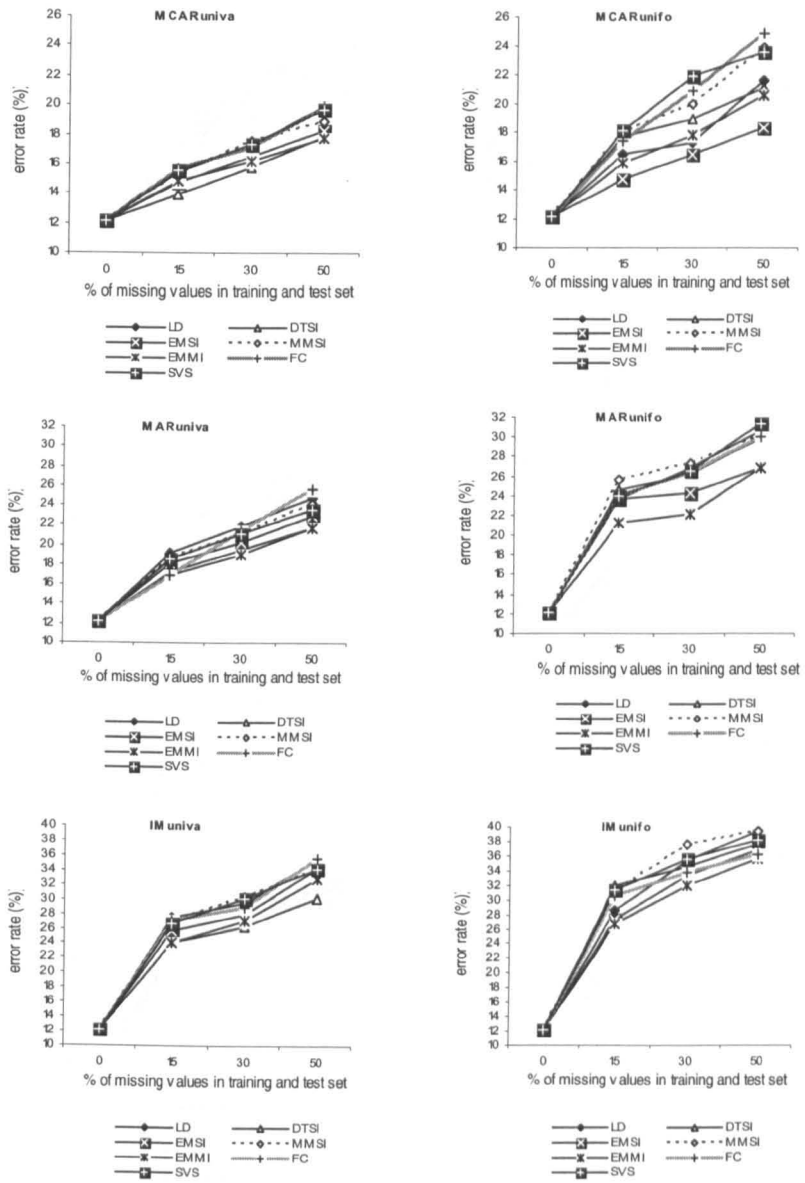

Fig. 4.10. Comparative results of methods for the letter dataset. A) MCAR*univa*, B) MCAR*unifo*, C) MAR*univa*, D) MAR*unifo*, E) IM*univa*, F) IM*unifo*

4.4.1.1.2 Results on a dataset with purely nominal attributes: kr-vs-kp

From Figure 4.11A, it follows that EMMI and FC are the best techniques for handling MCAR*univa* data. However, LD becomes more effective at higher levels of missing values.

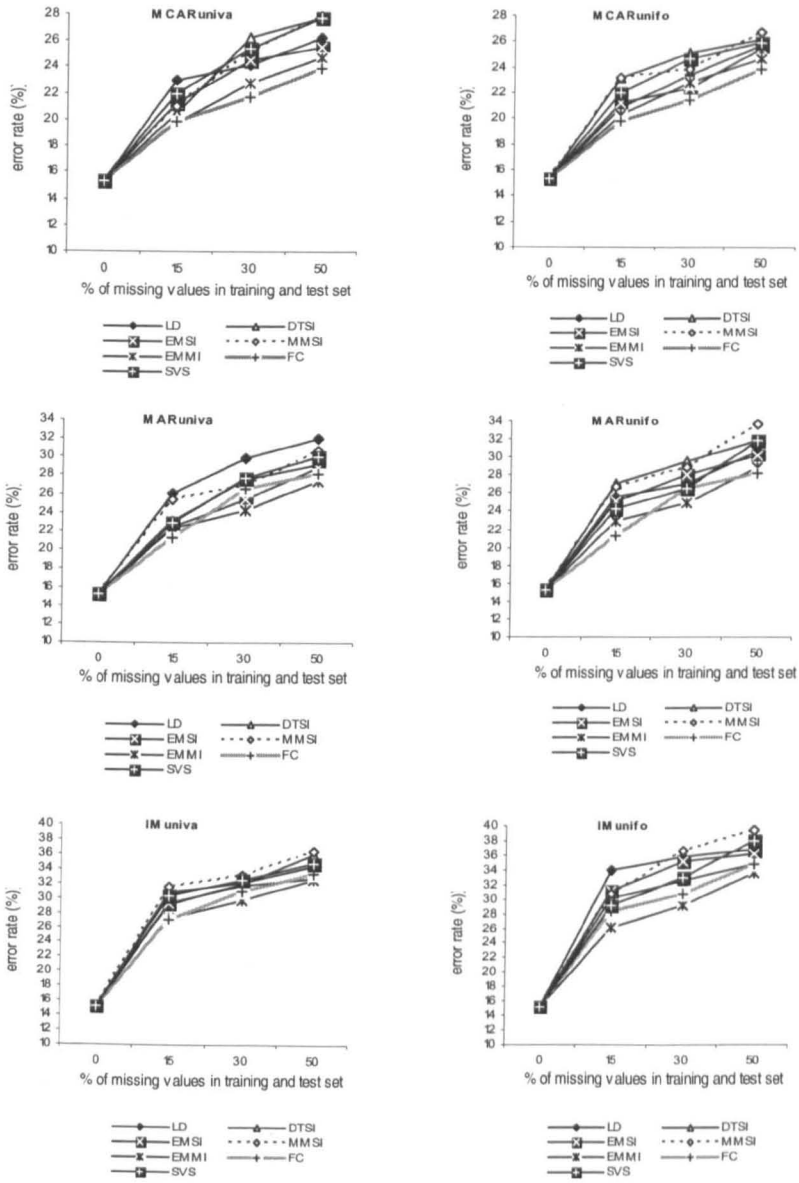

Fig. 4.11. Comparative results of methods for the kr-vs-kp dataset. A) MCAR*univa*, B) MCAR*unifo*, C) MAR*univa*, D) MAR*unifo*, E) IM*univa*, F) IM*unifo*

In the MAR*univa suite*, EMMI and FC show superior performances compared with the other methods (Figure 4.11C). For MAR*unifo* data FC outperforms EMMI, especially at higher levels of missing values (Figure 4.11D). The worst overall performance is by MMSI. For IM*univa* data, the worst performance is by MMSI and

LD (Figure 4.11E). The behaviour of methods for IM*unifo* data displayed in Figure 4.11F show good performances by FC and DTSI.

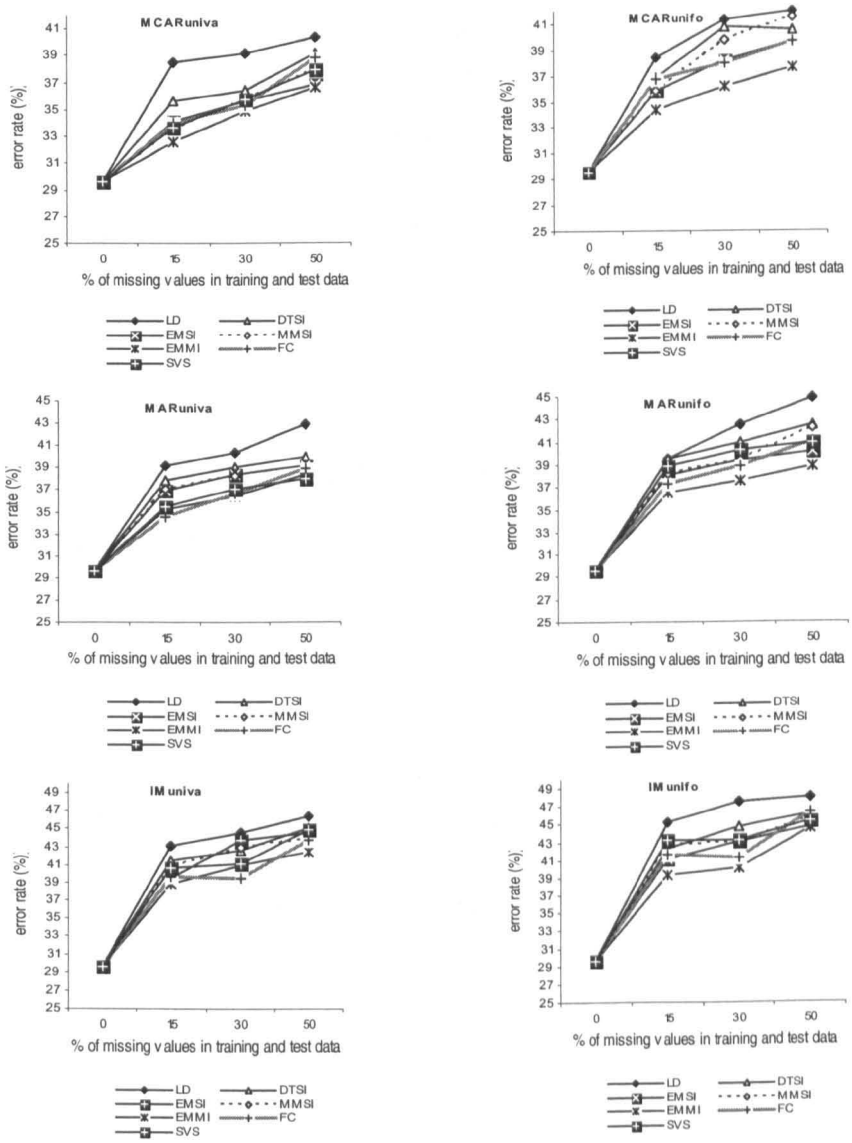

Fig. 4.12. Comparative results of methods for the german dataset. A) MCAR*univa*, B) MCAR*unifo*, C) MAR*univa*, D) MAR*unifo*, E) IM*univa*, F) IM*unifo*

4.4.1.1.3 Results on a dataset with mixed attributes: german

From Figure 4.12A, LD exhibits the worst performance for MCAR*univa* data. SVS exhibits one of the best performances for MCAR*unifo* data (Figure 4.12B). EMSI appears to be less effective than it was for the MCAR*univa* suite (Figure 4.12C). Results for MAR*unifo* show FC's performance becoming less effective as the amount of missing values increases (Figure 4.12D). From Figure 4.12E, EMMSI achieves one of the best performances for IM*univa* data. Results achieved by methods for IM*unifo* data (Figure 4.12F) show SVS becoming more effective as the proportion of missing values increases.

4.4.2 Overall Results - Incomplete Training Data Only

The excess error rates of each training method against three amounts of missing values are displayed in Figure 4.13. The excess error rates of each method of the introduced missing values are averaged over the 21 datasets. For both the missing data patterns and missing data mechanisms conditions, increases in error rates are associated with increases in proportion of missing values.

For MCAR*univa* data, EMMI has on average the best accuracy throughout the entire spectrum of amounts of missing values (Figure 4.13A). From Figure 4.13B, all the methods achieve bigger error rate increases for MCAR*unifo* data compared with MCAR*univa* data. In contrast to the missing-on-single attribute case, DTSI also achieves one of the biggest increases in error rate. In the MAR*univa* suite, the behaviour of methods is not different from the one observed in the MCARuniva case (Figure 4.13C). Figure 4.13D shows EMSI and FC achieving the second smallest error rate increases. CCSI becomes less effective as the proportion of missingness increases. The results in Figure 4.13E show bigger increases in error rates for all the methods for IM*univa* data. The results are otherwise nearly identical to those observed for MCAR*univa* and MAR*univa* data. The performance of all the methods, on average, worsens for IM*unifo* data (Figure 4.13F).

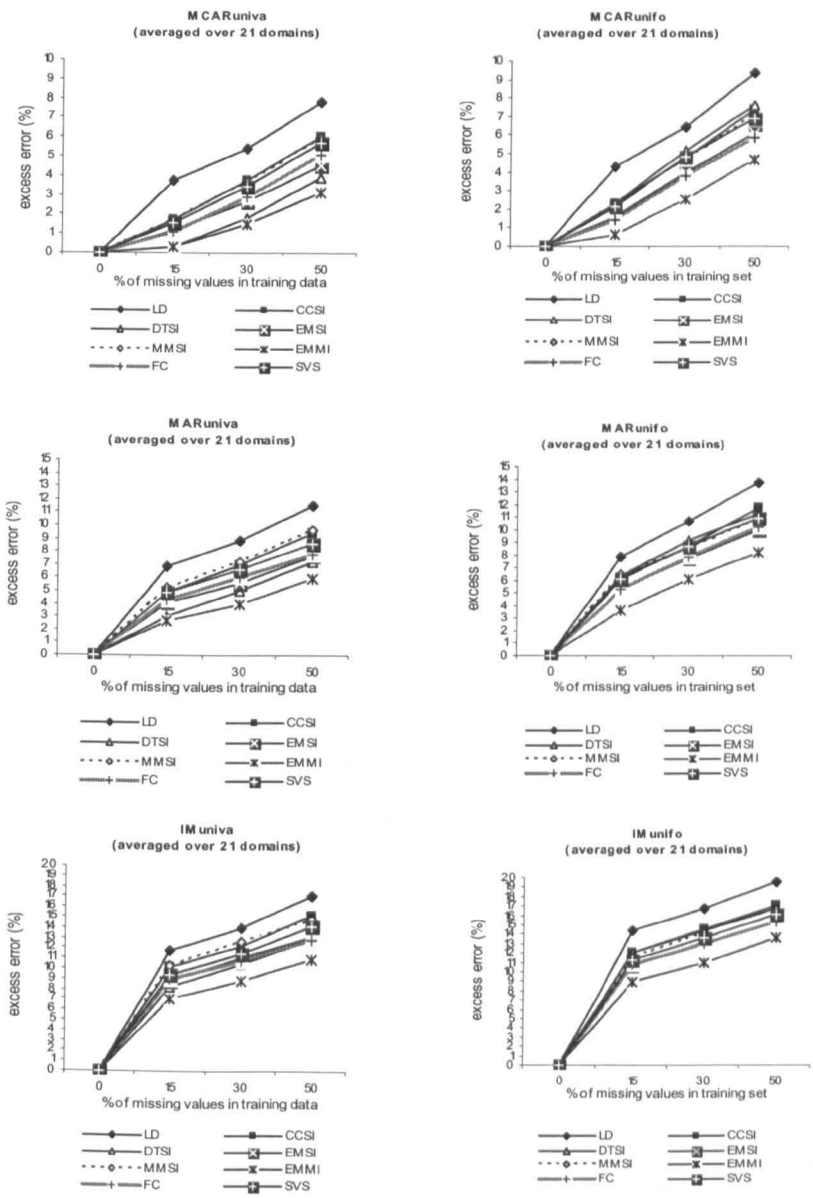

Fig. 4.13. Effects of missing values in training data on the excess error for methods over the 21 domains. A) MCAR*univa*, B) MCAR*unifo*, C) MAR*univa*, D) MAR*unifo*, E) IM*univa*, F) IM*unifo*

Main Effects

All the main effects (training methods, number of attributes with missing values, missing data proportions and missing data mechanisms), as displayed in Table 4.4 in the Appendix, were found to be significant at the 1% level of significance.

From Figure 4.14 it follows that EMMI is the overall best technique for handling incomplete training data. LD is the least accurate approach. Figure also 4.14 shows CCSI, DTSI, EMSI, MMSI, FC and SVS performing comparably among each other.

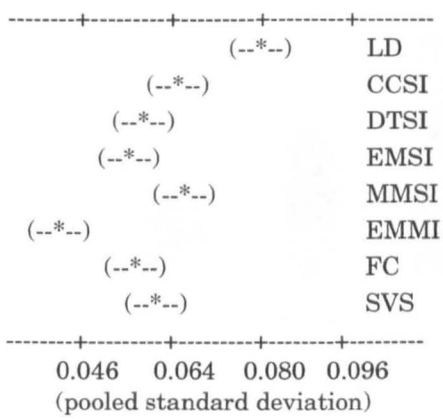

Fig. 4.14. Comparison for training methods: confidence intervals of mean error rates (*)

From Figure 4.15, it appears that missing values have a greater effect when they are distributed among all the attributes (with an error rate increase of about 9%) compared with when they are distributed in only one attribute variable (with an error rate increase of about 7%). The results for proportion of missing values in the training set show increases in missing data proportions being associated with increases in error rates (Fig. 4.16).

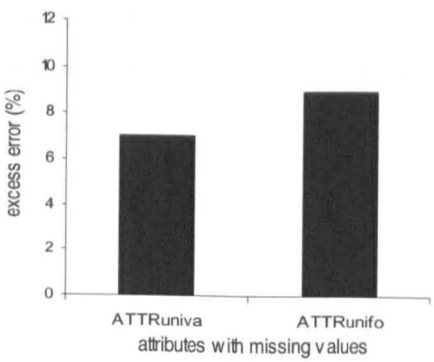

Fig. 4.15. Overall means for number of attributes with missing values in training set

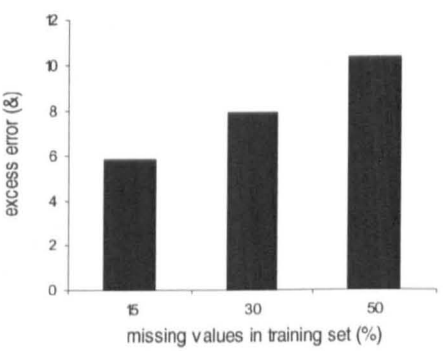

Fig. 4.16. Overall means for missing data proportions (training methods)

The results presented in Figure 4.17 show all the training methods performing worse under the IM condition than when data are MCAR or MAR. However, all the methods were able to handle MCAR data better than MAR.

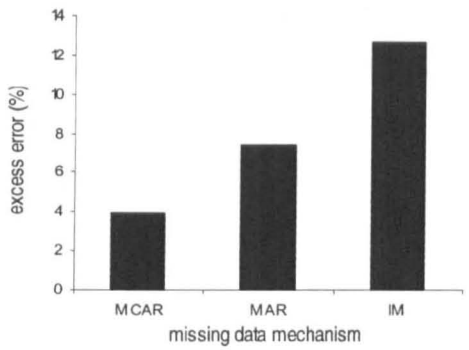

Fig. 4.17. Overall means for missing data mechanisms (training methods)

Interaction effects

The experimental results of the interaction effects for training methods follow a similar pattern to previous results for training and testing methods. In fact, all the two-way interaction effects that were statistically significant for training and testing methods are also significant for training methods (with the exception of the two-way interaction between the number of attributes with missing values and missing data mechanisms which is significant for the latter, as displayed in Figure 4.18). The interaction effect between the number of attributes with missing values and missing data mechanism show missing values having more impact when they are IM and are distributed among all attributes compared with when they are MCAR or MAR (Figure 4.18).

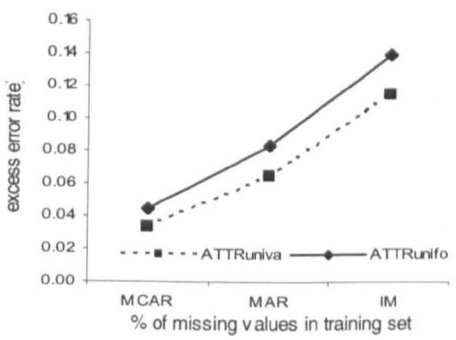

Fig. 4.18. Interaction between number of attributes with missing values and proportion of missing values

4.4.3 Overall Results – Incomplete Test Data Only

Figure 4.19 summarises the excess error rates of each testing method against three amounts of missing values. The error rates of each method of the introduced missing values are averaged over the 21 datasets.

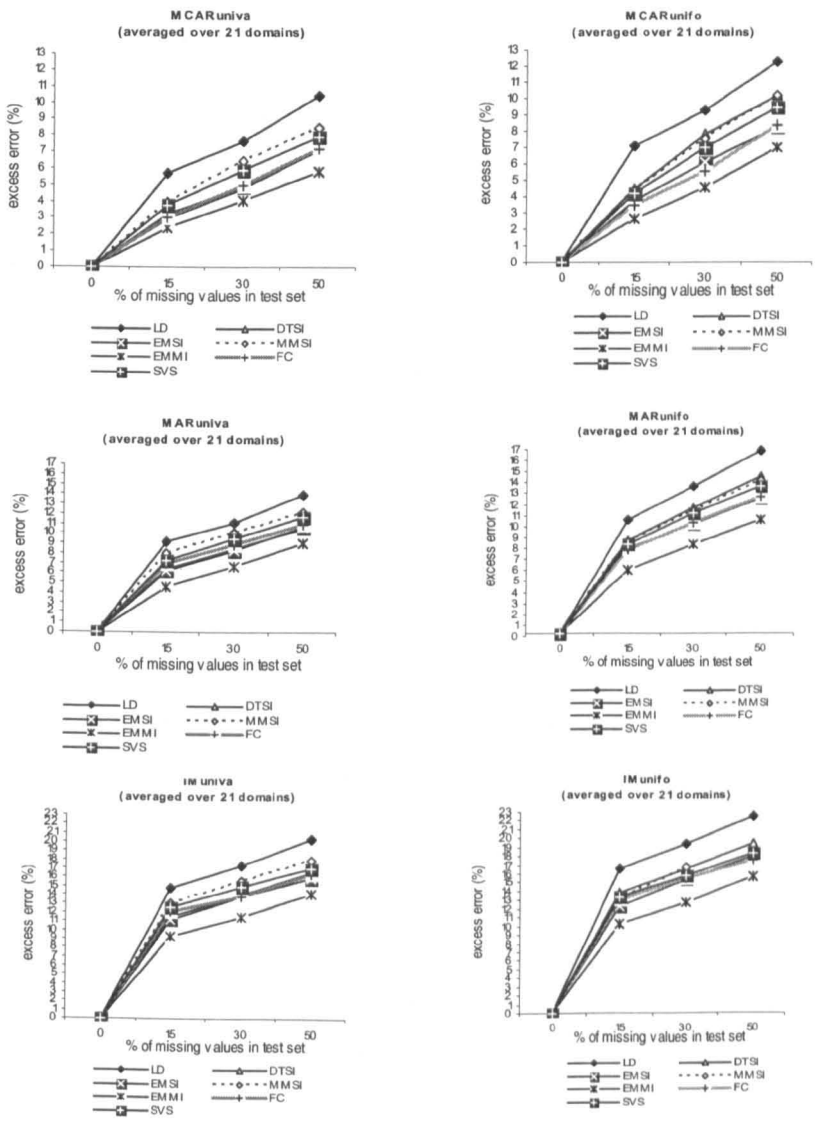

Fig. 4.19. Effects of missing values in test data on the excess error for methods over the 21 domains. A) MCAR*univa*, B) MCAR*unifo*, C) MAR*univa*, D) MAR*unifo*, E) IM*univa*, F) IM*unifo*

Once again, the error rate differences are relative to error rates at the control level (when there are no missing values), especially at lower levels of missing values. The performance of all the methods for handling incomplete test data only worsens compared with the same methods used for handling incomplete training data, i.e. the error rate increases for testing methods are a little bigger compared with error rate increases for training methods.

Figure 4.19A shows that EMMI has on average the best accuracy throughout the entire spectrum of amounts of missing values for MCAR*univa* data, closely followed by DTSI. From Figure 4.19B, SVS appears to be more effective for handling MCAR*unifo* data compared with MCAR*univa*. For MAR*univa* data, FC and SVS perform better than expected in particular situations compared with their performance as training methods (Figure 4.19C). For MAR*unifo* data, the pattern of results is similar to the one observed in the MCAR*unifo* suite (Figure 4.19D). Good performances are achieved by EMSI and FC while DTSI becomes less effective as the proportion of missingness increases for IM*univa* data (Figure 4.19E). EMMI is the most effective method for handling IM*unifo* data (Figure 4.19F). The differences in performance by methods appear to increase with increases in the proportion of missingness.

Main effects

Once again, all the main effects (testing methods, number of attributes with missing values, missing data proportions and missing data mechanisms) were found to be significant at the 1% level of significant (See Table 4.5 in the Appendix).

As it was the case with the results of training methods, EMMI is the best technique for handling incomplete test data (Figure 4.20). However, EMSI is replaced by FC as the second best testing method. LD exhibits the worst performance, followed by MMSI. All the methods (with the exception of LD and EMMI) appear to have on average comparable classification accuracy as testing methods.

The results of the other main effects (i.e. the number of attributes with missing values, proportion of missing values and missing data mechanisms) for testing methods are identical to the results achieved by training methods. The only

difference is that predictive error rates achieved by testing methods are slightly bigger than those achieved by training methods.

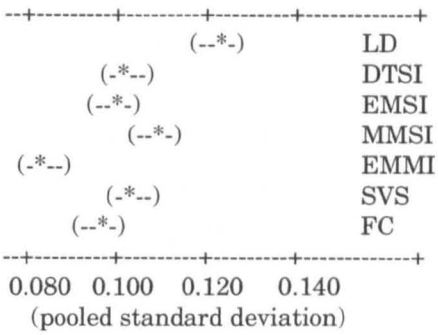

Fig. 4.20. Comparison for testing methods: confidence intervals for mean error rates

Interaction effects

All the two-way interaction effects that were found to be significant for training and testing methods, and later training methods, are also significant for testing methods at the 1% level. The key difference is that predictive accuracy rates achieved by testing methods are bigger than those achieved by training methods. Also, the interaction between the number of attributes with missing values and the proportion of missing values is not significant for testing methods as it was the case for training methods. The reader is referred to Table 4.5 in the Appendix for a detailed ANOVA results.

4.4.3.1 Supplementary Experiment and Results

Based on previous experimental results, which generated some interesting results with respect to some database characteristics (such as type of attributes, number of attributes, number of instances, number of classes), a subsidiary analysis for unbalanced data was carried out. The analysis was for assessing how each testing method is impacted by missing values on specific type of attribute variables. As regards to the type of datasets, a binary attribute variable which takes the value 1 if and only if the number of nominal attributes exceeds the number of ordered

attributes, and 0 otherwise, was created. In other words, the 'binary variable' is used to indicate whether or not the dataset is predominantly nominal. This was the case for six datasets (german, kr-vs-kp, anneal, LED 24, lymphography, and zoo). An additional complication is that the ANOVA is now non-orthogonal (unbalanced). However, an ANOVA procedure which can analyze the variance of unbalanced data was used for this analysis. The interested reader is referred to Miller (1997) and Kitchenham (1998) for a discussion on ANOVA for unbalanced data. In addition, only testing methods are considered for this experiment since previous results showed methods as more severely impacted when missing values are in test data than in training data.

All the main effects (testing methods, number of attributes with missing values, missing data proportions, missing data mechanisms and the 'binary attribute' were found to be significant at the 1% level of significance, as shown in Table 4.6 (a) in the Appendix. However, the main output table is now largely irrelevant, except for the binary attribute effect which was tested against the residual error. This is to avoid anomalies arising from the non-orthogonality among the effects. For each of the other effects, the appropriate error term from Table 4.6 (a) is used, involving an interaction with dataset nested within 'binary' attribute, i.e. each effect and its respective interaction with 'binary attribute' has its own error term. The results are presented in Table 4.6 (b) in the Appendix. In addition, the only interaction effects considered for this analysis were for each main effect against the 'binary attribute variable'. All the interaction effects were found to be not significant at the 1% level as shown in Table 4.6 (b) in the Appendix.

4.5 Discussion

The research questions asked which MDTs yielded the least amount of average error when using tree-based models.

The results of the simulation study show that the proportion of missing data, the missing data mechanism, the pattern of missing values, and the design of database

characteristics (especially the type of attributes) all have effects on the performance of any MDT.

The effects of missing data have been found to adversely affect DT learning and classification performance, and this effect is positively correlated with the increasing fractions of missing data.

Another point of discussion is the significance of having missing values in only one attribute, on the one hand, and allowing missing values in all the attributes, on the other hand. The idea was to see the impact of pattern over mechanism, or vice versa, at both lower levels and higher levels of missing values. Our results show the impact on the performance of methods being caused by the pattern and mechanism of missing values, especially at lower levels of missingness. However, as the proportion of missing values increases the major determining factor on the performance of methods is how the missing values are distributed among attributes. All methods yield lower accuracy rates when missing values are distributed among all the attributes (MCARunifo, MARunifo and IMunifo) compared with when missing values are on a single attribute (MCARuniva, MARuniva and IMuniva).

The worse performance achieved by methods are for IM data, followed by MAR and MCAR data, respectively. This is in accordance with statistical theory which considers MCAR easier to deal with and IM data as very complex to deal with since it requires assumptions that cannot be validated from the data at hand (Little and Rubin, 1987). In addition, in many settings the MAR assumption is more reasonable than the MCAR. In fact, an MAR method is valid if data are MCAR or MAR, but MCAR methods are valid only if data are MCAR. This could have attributed to the superior performance of EMMI (an MAR method) and the substantially inferior performance of LD (an MCAR method).

The results also show that the performance of methods depends on whether missing values are in the test or training set or in both the training and test sets. Training methods appear to achieve superior performances compared with testing methods. An explanation for such behaviour will be given later in the section.

With this experimental set-up, it is easy to say with conviction that from the eight current techniques investigated that EMMI is the overall best method for handling both incomplete training and test data. However, there are competitors like FC and DTSI which performed reasonably well. One important advantage of FC over DTSI is that it can handle missing values in both the training and test sets while DTSI struggles as a technique for handling incomplete test data and when all attributes have missing values. The heavily dependence of DTSI on strong correlations among attributes might have attributed to its poor performance as correlations among attributes for some of the datasets were not strong. However, DTSI performs better when missing values are on a single attribute – a very serious restriction. The results also indicate that LD is the worst method for handling incomplete data. In general, it can be seen that model-based methods have better performance than ad hoc methods. Furthermore, probabilistic methods seem to outperform non-probabilistic methods.

There are several dimensions on which learning methods of handling incomplete data using tree-based models can be compared. Also, combinations of methods for handling incomplete data while varying the number of attributes with missing values were not tried. However, prediction accuracy rates of estimation methods like the EMMI were very impressive. The improvement in accuracy of EMMI over single imputation methods (CCSI, DTSI, EMSI and MMSI) and other methods (LD, SVS and FC) could be as a result of a reduction in variance resulting from averaging the number of trees like is done in bagging (Breiman, 1996). Even though EMMI emerges as the overall best of the eight techniques, it has come under fire by critics claiming that proper imputations, necessary for valid inferences, are difficult to produce, especially in data where multiple factors are deficient (Schafer, 1997), and even then EMMI is biased in some cases (Robins and Wang, 2000). Onother argument against EMMI is that it is much more difficult to implement than some of the techniques mentioned. One potential problem that was encountered in this research is convergence of the EM algorithm (Wu, 1983), especially for big datasets and datasets with more than 30 attribute variables.

The results also show that the performance of methods depends on whether missing values are in the test or training set or in both the training and test sets. When looking at the overall performance of methods, training methods appear to achieve superior performances compared with testing methods. However, in terms of relative performance, they seem to be about the same.

Each dataset might have more or less its own favourite techniques for processing incomplete data. However, most of our dataset results are similar to one another. Several factors contribute here: the methods used; the different types of datasets; the distribution of missing values among attributes, the magnitude of noise, distortion and source of missingness in each dataset.

There was evidence from our results that how well a method performs depends on the dataset one is analysing. In fact, all methods were able to handle datasets with numerical attributes better compared with datasets with only nominal or a mixture of both nominal and numerical attributes.

For small datasets all the techniques seemed to work well with the estimation methods, especially EMMI which performed better than both the other estimation methods and machine learning methods. One the other hand, LD gave the worst performance for small datasets. This seems to be logical since when using LD you tend to lose a lot of information, especially at higher levels of missing values. However, for bigger datasets, LD was equally effective compared with methods like CCSI and MMSI and outperformed them in other situations. The effectiveness of LD probably stems from the fact that deletion techniques result in data matrices that mirror the true data structure. When data are systematically missing from a study, the imputation techniques create a "reproduced" data matrix with a structure somewhat different from that of the true data matrix.

Following EMMI as the second best overall method for handling both incomplete training and test data is FC. In general it can be seen that probabilistic methods have better performance than non probabilistic methods. However, one important disadvantage of FC, just like EMMI, is that it takes a long time processing (especially big trees) due to the way it handles missing values. Due to its reliance on

the number of branches to do the calculation simultaneously, if K branches do the calculation, then the CPU time spent is K times the individual branch calculation.

Another good performance was by CCSI especially for datasets with only a few classes and with a mixture of both numerical and qualitative attributes. The worse performance by CCSI was observed for small datasets with many classes. Hence, our results suggest that methods such as CCSI for handling incomplete data work better for datasets where the response variable has a few number of classes and a few attributes. However, these attributes should be highly correlated to the class variable. One serious limitation of CCSI is that it can only handle incomplete training data but not incomplete test data.

Of all the single imputation methods, EMSI seems to perform the best for datasets with numerical attributes. However, despite its strengths, EMSI suffers (like all the single imputation methods) from biased and sometimes inefficient estimates. DTSI and MMSI achieved good results when missing values were on categorical attributes. For DTSI, this was the case when the missing values were in only one attribute while MMSI gave good results generally and in some cases was a good method for datasets with mixed attributes. Overall, the differences in results between single imputation methods are relatively small. The similarity of results begs an important question: when and why should we choose one single imputation method over the other?

Like DTSI, for all the datasets where the correlations among attributes were found to be quite high, SVS (which relies heavily on strong concordance between a primary splitter and its surrogate(s)), achieved good results. However, SVS also struggles when missing values are distributed among all the attributes. In fact, for a few datasets SVS collapsed completely when an instance was missing all the surrogates. However, some strategies when simulating the missingness among the attributes were used when the technique collapsed. In addition, the performance of SVS improves as a method for handling incomplete test data compared with handling incomplete training data.

Another point of discussion is why missing values are more damaging when they are in the test sample than training sample. If you have a lot of training data then missing values do not make much impact on the parameter estimates but missing data in the test set refers to only individual cases. That is, the training data yield statistical summaries, but the test data are concerned with individuals. In other words, missing data will tend to cancel each other out when training the model. On a new test case, the investigator must still suffer accuracy affects though, inevitably. What is really happening here is that the increased error in test cases is to be expected and the significantly reduced error when training is a pleasant surprise and this is due to the averaging.

Furthermore, it is worthwhile mentioning that the performance of some methods could have been slightly affected by other factors like errors in some datasets. For example, the Pima Indians diabetes database had quite a number of observations with "zero" values, which are most likely to indicate missing values although the data was described as being complete. Nonetheless, the prediction that the impact of certain types of missing data mechanisms on both the testing and training cases should differ by dataset, by mechanism and by the proportion of missing values is confirmed.

Some results from Section 4.2 support previous findings in the literature and other results extend the literature. The relative superiority of model-based methods over ad hoc methods is consistent with past results (Kim and Curry, 1977; Little and Rubin, 1987; Rubin, 1987).

Overall, the performance of each MDT under more complex forms of systematic missingness is unknown and likely to be problematic (Little and Rubin, 1987). Systematic missingness in this simulation was always based on the variables that were in the model rather than unmeasured variables or combinations of variables. In addition, it is impossible, in practice, to demonstrate whether data are MAR versus IM, because the values of the missing data are not available for comparison. IM is still a problem for the methods reviewed here.

Chapter 5

More on the Problem of Building Trees Using Incomplete Vectors and Classifying Incomplete Vectors Using Trees

5.1 Introduction

Learning with incomplete data has been a challenge in both the Machine Learning and Statistics communities. In a situation when there are no missing values (unknowns) in either the training set or test sets, the tree construction and classification processes are carried out in the expected manner. However, if missing attribute values exist in either the training set or test set or in both sets, the tree building or classification processes are affected.

This chapter will outline the proposed procedure for constructing tree classifiers using incomplete training vectors and classifying incomplete vectors, i.e. a technique that can be used to handle missing attribute values in both training and test data is developed. There are few techniques (especially those that utilises machine learning) algorithms) that can handle missing attribute values in both training and test data. There are several important aspects of handling incomplete training and test data using DTs which the proposed procedure casts light on. Perhaps the most important of these is the failure of existing methods to handle IM data well, as shown in the experiments in Chapter 4. The goal is to produce a technique that can be used to handle any type of attribute with missing values and be able to deal with any of the three missing data mechanisms effectively, especially IM data. As with Quinlan's approach, the proposed procedure regards "missing" as another category but the analysis and resulting algorithms are different from those of Quinlan. The key difference is that numeric or ordered attributes are not quantized or discretized first

before using this strategy. Instead, the missing values constitute special values that are assigned a place in the ordering that yields the best split. The place is generally different in different nodes of the tree. The method uses three prospective binary or two-way splits; two of which accommodate the "missingness" of the data along with actual values and one "missingness" and "non-missingness" of the data as a dichotomy. In other words, the proposed procedure uses "missingness" as a pseudo value which is incorporated with the other attribute values. The binary split that maximises the impurity criterion that had been used to grow the tree is the one chosen, thus, the best variable. This pseudo-value makes the proposed technique a more practical and powerful approach.

The usefulness of the proposed procedure from the point of view of its effect or tolerance to incomplete training and test data is investigated experimentally. In particular, the focus is on the 15%, 30% and 50% levels of missing values, having missing values on only one attribute variable and having missing values distributed among all attributes, individually; and three different mechanisms of missing values that are known to distort data (MCAR, MAR and IM). In addition, it is expected that the proposed method will handle IM data more effectively compared with existing methods since it considers "missingness" as important when determining the best split when growing the tree or when classifying an unknown instance. In other words, the proposed approach could be considered to be stronger on the philosophy and foundations for anything we have that is IM.

This remainder of this chapter is organised as follows: In Section 5.2 the proposed method, against the background of the current techniques is introduced. In Section 5.3 the experimental methodology is briefly explained. Results of simulation studies showing the performance of the proposed method against current techniques are contained in Section 5.4. Some discussion of the results in Section 5.5 closes the chapter.

5.2 Building Decision Trees Using Incomplete Training Vectors and Classifying Incomplete Vectors – Proposed Procedure

DTs are generally learned by means of a top down growth procedure, which starts from the root node and greedily (picks the best attribute and never looks back to consider earlier choices) chooses a split of the data that maximises some cost function, usually a measure of the 'impurity' of the sub-samples implicitly defined by the split. The estimation criterion in the decision tree algorithm is the selection of an attribute to test at each internal node in the tree. The goal is to select the attribute that is most useful for classifying examples. The most common measure is the statistical property called the information gain or transmitted information that measures how well a given attribute separates training instances according to their target classification (Quinlan, 1986; 1993). However, there have been a number of alternative measures for selecting attributes (the reader is referred to Section 2.2). For the purposes of the proposed technique, the GINI index of diversity (Breiman *et al.*, 1984), which measures the 'impurity' of an attribute with respect to the classes shall be used. In fact, the GINI index was also used as the goodness-of-split criterion for all experiments carried out in this thesis.

The proposed approach, which is now going to be called missingness-incorporated-in-attributes (MIA), follows the following specific form.

5.2.1 Learning Phase

An unknown (missing) value is considered an additional attribute value. Hence the number of values is increased by one for each attribute that depicts an unknown value in the training or test set.

If X_n is an ordered or numeric attribute variable with unknown attribute values, the proposed approach searches essentially over all possible values of x_n for binary splits of the following form:

$$\begin{aligned}
&(i)\ \textit{Split } A\text{: } (X_n \le x_n \text{ or } X_n = \textit{missing}) \text{ versus } (X_n > x_n)\\
&(ii)\ \textit{Split } B\text{: } (X_n \le x_n) \text{ versus } (X_n > x_n \text{ or } X_n = \textit{missing})\\
&(iii)\ \textit{Split } C\text{: } (X_n = \textit{missing}) \text{ versus } (X_n = \textit{not missing}).
\end{aligned} \tag{5.1}$$

The idea is to find the best split from the candidate set of splits given above, with the goodness of split measured by how much it decreases the impurity of the sub-samples.

If X_n is a nominal attribute variable (i.e., a variable that takes values in an unordered set), the search is over all splits of the form:

$$\begin{aligned}
&(i)\ \textit{Split } A\text{: } (X_n \in Y_n \text{ or } X_n = \textit{missing}) \text{ versus } (X_n \notin Y_n)\\
&(ii)\ \textit{Split } B\text{: } (X_n \in Y_n) \text{ versus } (X_n \notin Y_n \text{ or } X_n = \textit{missing})\\
&(iii)\ \textit{Split } C\text{: } (X_n = \textit{missing}) \text{ versus } (X_n = \textit{is not missing})
\end{aligned} \tag{5.2}$$

where Y_n is the splitting subset at node n.

When the training set did not have any missing values for some variables, the above reduces to using a standard tree split for such variables, i.e., $X \le x_n$ versus $X_n > x_n$ (for an ordered attribute variable) or $X_n \in Y_n$ versus $X_n \notin Y_n$ (for a categorical attribute variable).

The standard algorithm for feature selection and the proposed algorithm for feature selection with unknown (missing) attribute values when using decision trees are displayed in Figures 5.1 and 5.2, respectively.

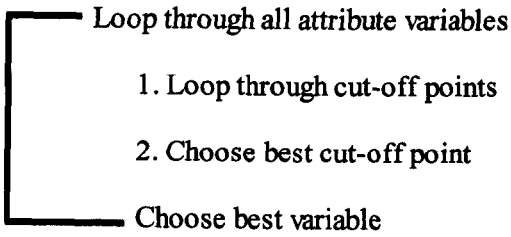

Figure 5.1 Standard Algorithm for feature selection

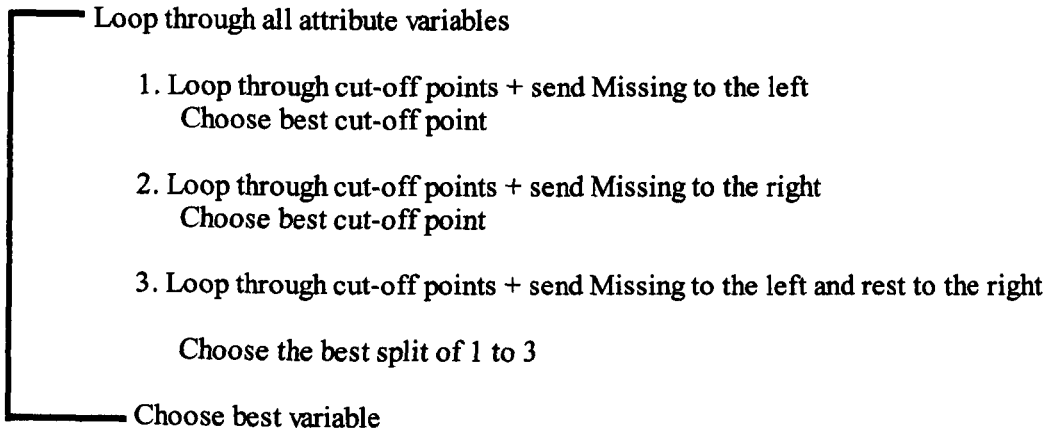

Figure 5.2 New Algorithm for feature selection with unknown attribute values

5.2.2 Classification Phase

The algorithm works in the same way to determine the outcome of the test when classifying a new instance, and given that at that particular internal node there are attribute values missing as it was the case with learning. If the unseen instance is regular (without any unknown attribute value) then the classification is carried out the traditional way. However, if an instance involves one or more unknown values, then the algorithm tries in turn all the three binary splits and selects the best split. The split chosen determines the number of instances branching on each path at a particular node for which a value is missing.

5.2.3 Illustration

To further illustrate the operation of the proposed procedure, consider the learning task represented by training instances with unknown (missing) attribute values given as an artificial dataset in Table 5.1.

Table 5.1 Artificial dataset with missing values (?) on attributes A_1 and A_3

Attribute 1 (A_1)	Attribute 2 (A_2)	Attribute 3 (A_3)	Class
?	936	9	1
1	1168	12	1
4	5117	27	1
1	902	?	2
4	1495	12	1
1	10623	30	1
4	1935	?	1
?	1424	12	1
1	6568	?	1
4	1413	12	1
4	3074	9	1
?	3835	30	1
1	5293	27	2
3	1908	?	2
4	3342	?	1
?	932	6	1
1	3104	18	1
3	3913	36	1
?	3021	24	1
4	1364	12	1
2	625	?	1
1	1200	12	1
?	707	12	1
4	2978	24	1
?	4657	15	1
4	2613	?	1
2	10961	48	2
1	7865	12	2
4	1478	9	2
1	3149	?	1
?	4210	36	2
4	2507	9	1
4	2141	12	1
2	866	?	1
4	1544	18	1
1	1823	24	2
?	14555	6	2
2	2767	?	2
4	1291	12	1
?	2522	21	1

Here the target attribute (also called class attribute), which can have values 1 or 2 for 40 instances, is to be predicted based on attributes (A_1, A_2, A_3) of the class in question. In instances 1, 8, 12, 16, 19, 23, 25, 31, 37 and 40, the available data has missing values for attribute A_1, while attribute A_3 has missing values for instances 5 ,7, 9, 12, 15, 20, 26, 30, 35 and 40. In other words, each attribute has ten missing values. A_2 is the only attribute with non-missing values.

As mentioned earlier, the proposed algorithm differs from the other algorithms in creation of this kind of DT in that it will always have one more choice per node. In addition, the DT is constructed by allowing 'missing' to be a possible choice on the decision tree.For an example, if split *A* from Equation 5.2 is considered as the best split for A_1 and split *C* from Equation 5.1 as the best split for A_3 then the resulting DT constructed using the dataset presented in table 5.1 is illustrated in Figure 5.3A. Figure 5.3B displays a decision tree constructed if split *C* is the chosen as the best split for A_1 with split *B* chosen as the best split for A_3. Note that in some situations the splitting criterion chosen could be the same for both the attributes with missing values.

Note that the proposed procedure is different from Quinlan's approach of handling unknown attributes values by considering 'missing' as another category. For example, with two binary variables, adding 'missing' as another category takes them up to trinary variables. Also, Quinlan's approach works well with categorical attributes. Continuous attributes are first discretized or quantized.

For the proposed strategy, 'missingness' is used as a pseudo-value and then carry out a binary splitting procedure to determine the best split at that particular node with missing attribute values. The proposed missing value strategy reportedly works fine for continuous attributes as well. Consider a fixed candidate split defined for non-missing values of a continuous attribute. Suppose the split would create two (2) branches. Missing values could be assigned to any one of the two branches. If an additional branch is allowed, the missing values could be assigned to a new branch that would contain no non-missing values.

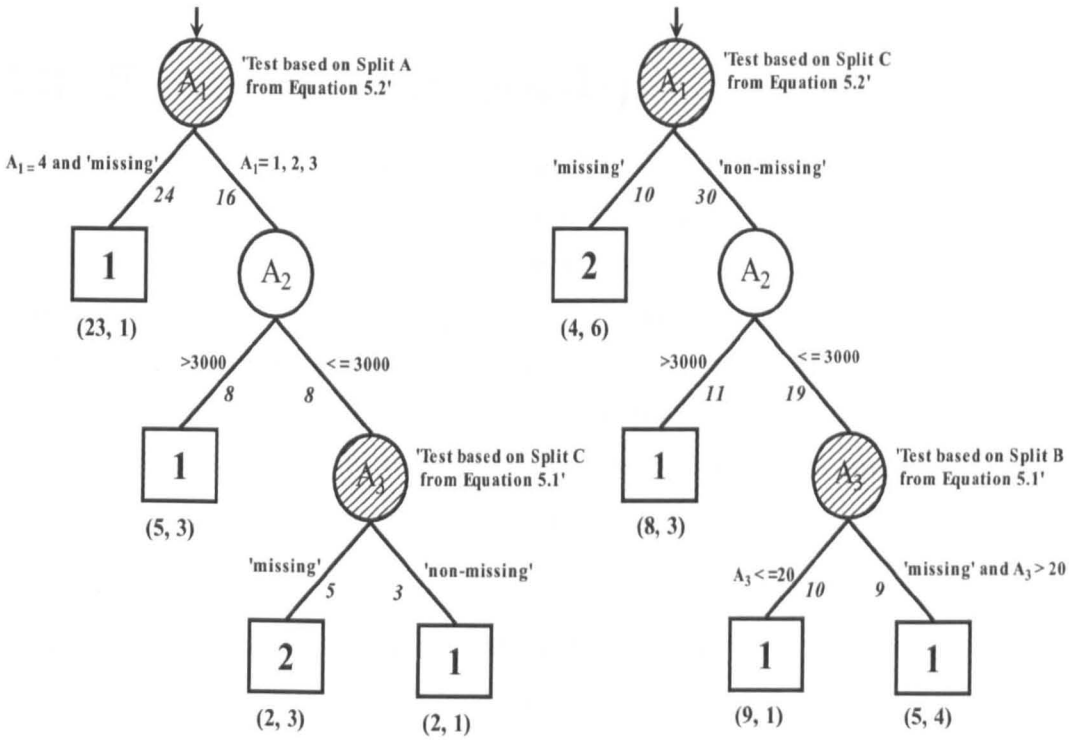

Figure 5.3. An artificial example of a simple binary decision tree which allows 'missing' to be a possible choice on the tree. A) Splitting criteria A and C from Equations 5.2 and 5.1, respectively, are used, B) Splitting criteria C and B from Equations 5.2 and 5.1, respectively are used

Note: Figures in brackets are the number of instances in each terminal or leaf node for class 1 and 2, respectively. Figures in italic represent training data instances that branch either to the right or to the left of each internal node at each respective cut-off point.

The proposed procedure evaluates each of these 2 or 2+1 revisions of the original candidate split. If C candidate splits on non-missing values into 2 branches are considered, then C times 2 (or 2+1) total splits are evaluated when including missing values. In other words, the splitting rule could assign missing values to anyone of the two branches. Cut-off candidates are done excluding the missing values. However, evaluation of the cut-off points are done including the missing values. Given a candidate cut-off, first try assigning missing values to the first branch, and then try the second branch. Finally assign, all missing values to the first branch and the non-missing values to the second branch.

5.3 Experimental Set-Up

In order to access the capabilities of each of the proposed and current methods, it is important to test them on several datasets. This will also aid determining the techniques strengths and weaknesses. Among the two current methods, EMMI (a statistical algorithm) has been employed as a baseline procedure since it achieved the highest accuracy rates in experimental results in the previous chapter. The FC strategy was also chosen as a comparator since not only did it achieve the second best overall performance in our previous experiments but is one of the major families of machine learning algorithms that can handle missing attribute values in training data and test data.

Each method was run for the two conditions which involve the number of attributes with missing values; four levels of missing data proportions in both the training and test sets and three missing data mechanisms. All combinations were tested on 21 datasets (already described in Section 4.3.1), and the scenario has been executed five times for each combination. Hence, the findings presented in this section are based on a total of 5670 conditions. Otherwise, the experiments have the same details as in the previous experimental section.

5.4 Experimental Results

Experimental results on the effects of current and proposed methods for handling incomplete training data and test data on predictive accuracy using DTs are described.

The results are presented in two parts. The first part compares the performance of three different approaches for building trees from incomplete vectors and classifying incomplete vectors using trees, looking at the overall results of each method, averaged for all twenty one datasets. The performance of the three methods on selected individual datasets is compared in the second part. These individual datasets are those where some interesting trends and results achieved by the three methods emerged.

5.4.1 Overall Results - Current Vs. New Methods

Figure 5.4 summarises the overall excess error rates for current and proposed training methods against three proportions of missing values. The excess error rates of each method of the introduced missing values are averaged over the 21 datasets.

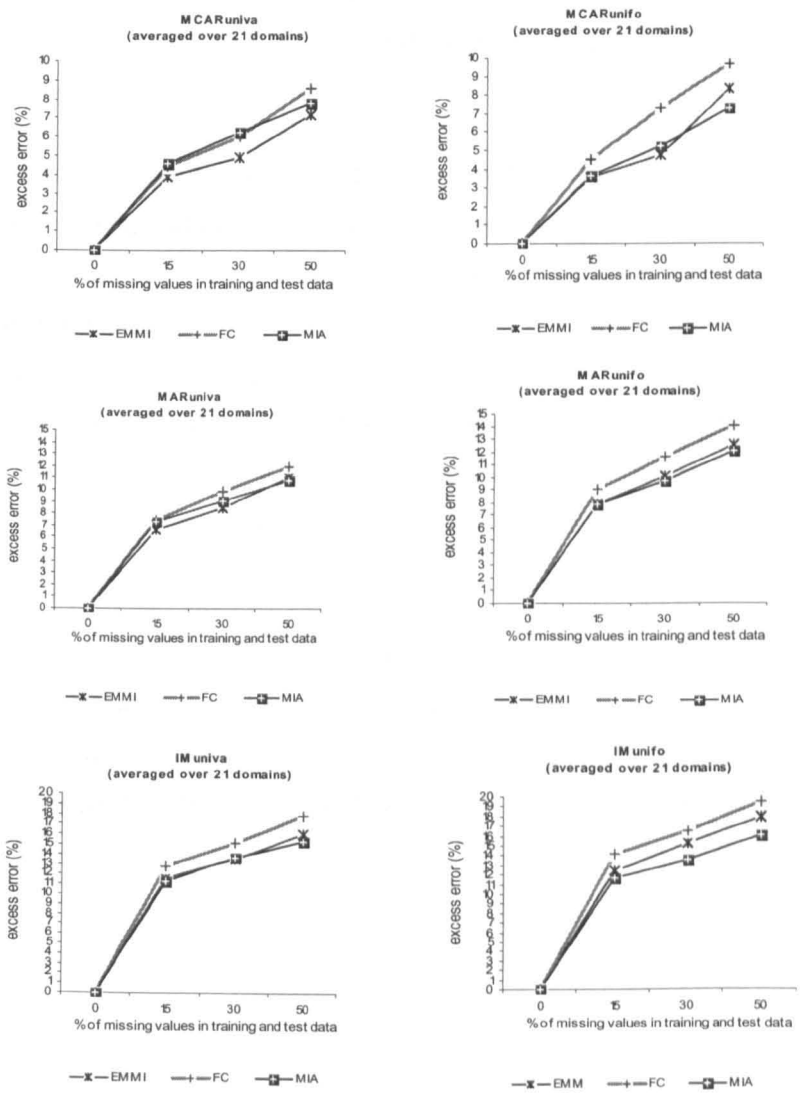

Fig. 5.4. Effects of missing values in training and test data on excess error for current and proposed testing methods. A) MCAR*univa*, B) MCAR*unifo*, C) MAR*univa*, D) MAR*unifo*, E) IM*univa*, F) IM*unifo*

From Figure 5.4A it is noticeable that when both training and test data are incomplete due to the MCAR*univa* mechanism, EMMI performs better than MIA. For MCAR*unifo* data MIA achieves the best performance at most levels of missing values (Figure 5.4B). The results displayed in Figure 5.4C show a poor performance by FC. Results for MAR*unifo* data follow a slightly similar pattern to the one observed for MAR*univa* data (Figure 5.4D). In this suite of IM*univa* experiments, the best performance at all levels is by MIA (Figure 5.4E). In the IM*unifo* case, as expected, MIA performs better than both EMMI and FC (Figure 5.4F).

Main Effects

All the main effects (existing and new training and testing methods, number of attributes with missing values, missing data proportions and missing data mechanism) were found to be significant at the 1% level as shown in Table 5.2 in the Appendix. Results for overall means for the main affects are similar to results from the previous experiments. That is, missing values have more impact when they are distributed among all attributes than when they occur in only one attribute. Also, increases in missing data are associated with increases in predictive error. Furthermore, bigger error rates are achieved by methods for IM data compared with MCAR or MAR data.

The performance of existing and proposed methods for handling incomplete training and test data is summarised in Figure 5.5. The best overall method for handling incomplete training and test data using decision trees is EMMI, closely followed by MIA and FC, respectively. However, the MIA and FC methods do not appear to be significantly different.

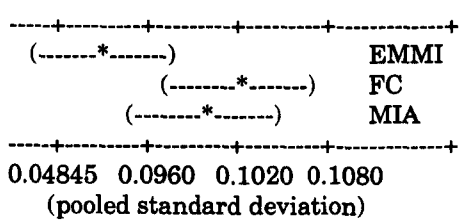

Fig. 5.5. Comparison for methods: confidence intervals of mean error rates

Interaction Effects

All the two-way interactions that were found to be significant in previous experiments for training and testing methods are also significant for this experiment (with the exception of the interaction between attributes with missing values and missing data mechanisms, which was not significant for the former). Figure 5.6 plots the interaction effect between number of attributes with missing values and missing data mechanisms which suggests that all the methods are severely impacted by missing values when they are distributed among all the attributes and are informatively missing than when they randomly missing. Also, the two lines for MCAR and MAR are almost parallel indicating no method being particularly less or more affected than others by these two missing data mechanisms.

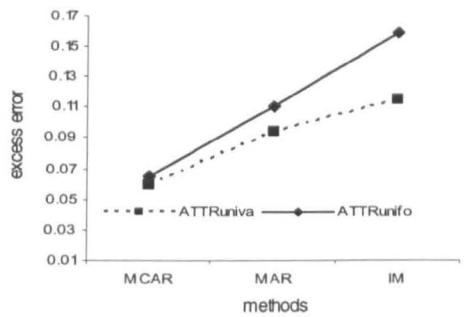

Fig. 5.6. Interaction between number of attributes with missing values and missing data mechanisms

5.4.2. Results for Individual Datasets – Current Vs. New Methods

Experimental results presented in this section illustrate specific deviations from the overall results of the effectiveness of the proposed method against the current methods for building and classifying unknown instances using trees and given incomplete data on different database characteristics. As in previous experiments, the accuracy of MIA is explored relative to that of EMMI and FC on individual datasets with purely numerical attributes, purely nominal attributes, and mixed attributes. The results report error rate of each method and are analyzed from the perspective of each of the input data characteristics.

5.4.2.1 Results on a dataset with purely numerical attributes: letter

For the letter data problem, the effects of missing values on classification accuracy are summarised in Figure 5.7A.

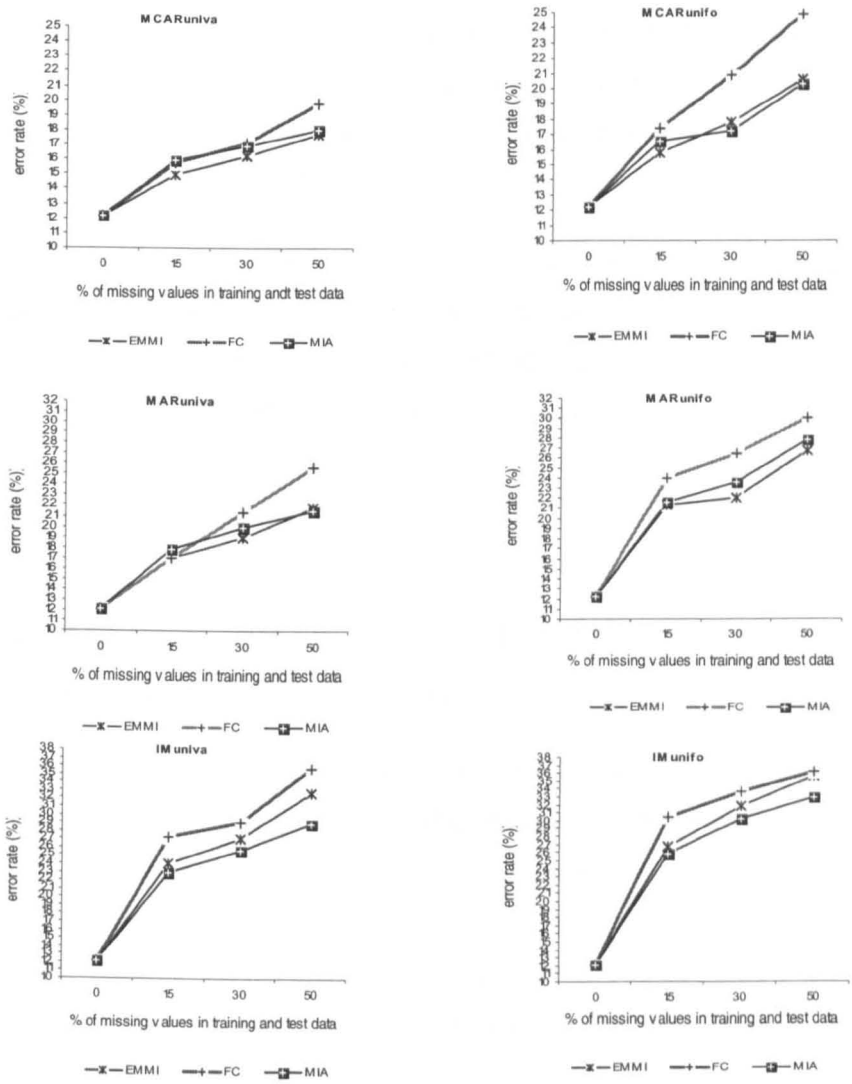

Fig. 5.7. Comparative results of current and proposed methods for the letter dataset. A) MCAR*univa*, B) MCAR*unifo*, C) MAR*univa*, D) MAR*unifo*, E) IM*univa*, F) IM*unifo*

In terms of tolerating MCAR*univa* data, EMMI performs slightly better than both MIA and FC. For MCAR*unifo* data, MIA yields the best performance (Figure 5.7B). Results for MAR*univa* data indicate EMMI and MIA exhibiting higher performances (Figure 5.7C). EMMI exhibits a very good performance for MAR*unifo* data (Figure 5.7D). The impact of IM*univa* data shows MIA outperforming both EMMI and FC. Also, the difference in performance between MIA and EMMI is now particularly obvious, especially at higher levels of missing values. In this suite of IM*unifo* experiments, the behaviour of the methods, shown in Figure 5.7F, is not different from the one observed in the IM*univa* case.

As expected, for this kind of dataset it appears that MIA handles this dataset quite well, especially for IM data (as expected) and MCAR (rather surprisingly). Also, the superior performance of MIA to EMMI for MAR data is rather surprising since one of the assumptions of EMMI requires that the data be MAR. In addition, this kind of data has purely numerical attributes with a reasonable number of attributes, which EMMI would normally handle quite well.

5.4.2.2 Results on a dataset with purely nominal attributes: kr-vs-kp

From Figure 5.8A the overall best performance for MCAR*univa* data is by MIA, with EMMI achieving lower accuracy rates. Results for the MCAR*unifo* data show MIA yielding the best performances (Figure 5.8B). Once again, EMMI is less effective for this condition. In the MAR*univa* suite, EMMI starts becomes more effective as the percentage of missing values increases (Figure 5.8C). The results for the MAR*unifo* suite, reported in Figure 5.8D, are almost similar to the one already observed in the MCARunifo suite. The behaviour of methods in the IM*univa* suite shows MIA outperforming both EMMI and FC (Figure 5.8E). In the IM*unifo* case, the behaviour of methods is similar to the one observed for IM*univa* data (Figure 5.8F). However, all the methods exhibit bigger error rates for the IM*unifo* data compared with IM*univa* data.

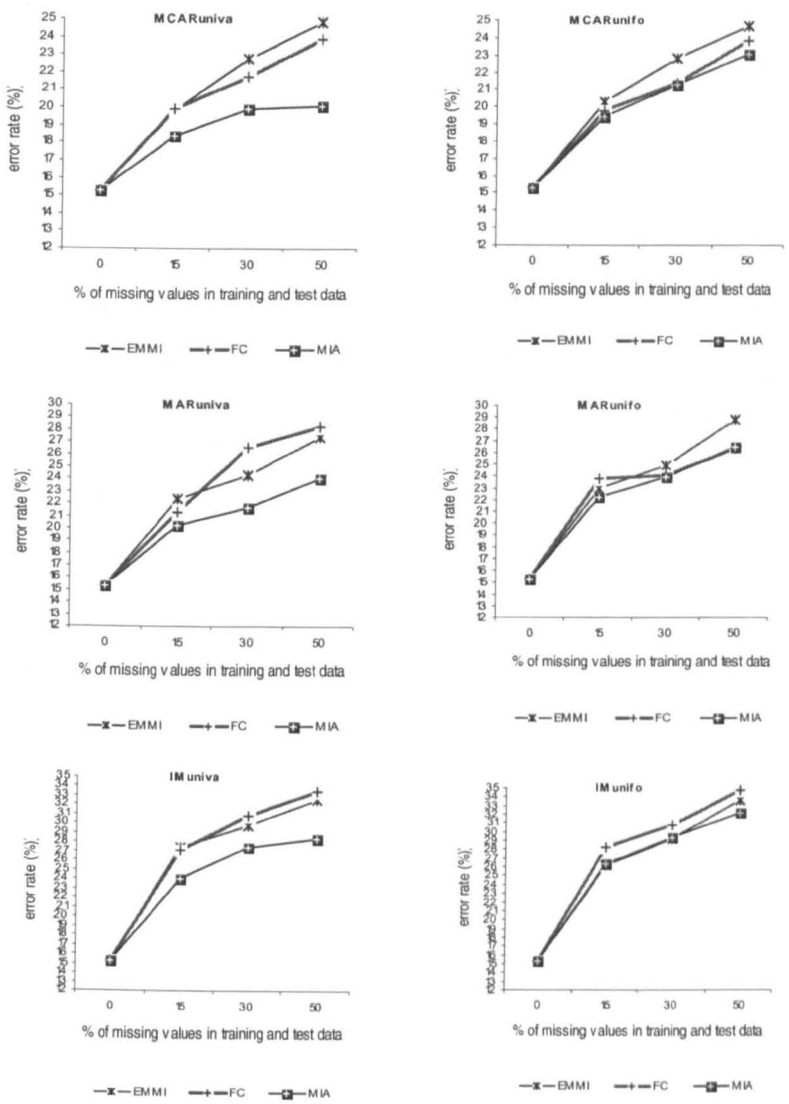

Fig. 5.8. Comparative results of current and proposed methods for the kr-vs-kp dataset. A) MCAR*univa*, B) MCAR*unifo*, C) MAR*univa*, D) MAR*unifo*, E)IM*univa*, F) IM*unifo*

5.4.2.3 Results on a dataset with mixed attributes: german

From Figure 5.9A it appears that for the MCAR*univa* mechanism, MIA performs slightly better than the other methods. For MCAR*unifo* data, the results show MIA, once again, achieving the best results (Figure 5.9B). From Figure 5.9C, results achieved by methods for MAR*univa* data are similar to results for MCAR*univa* data.

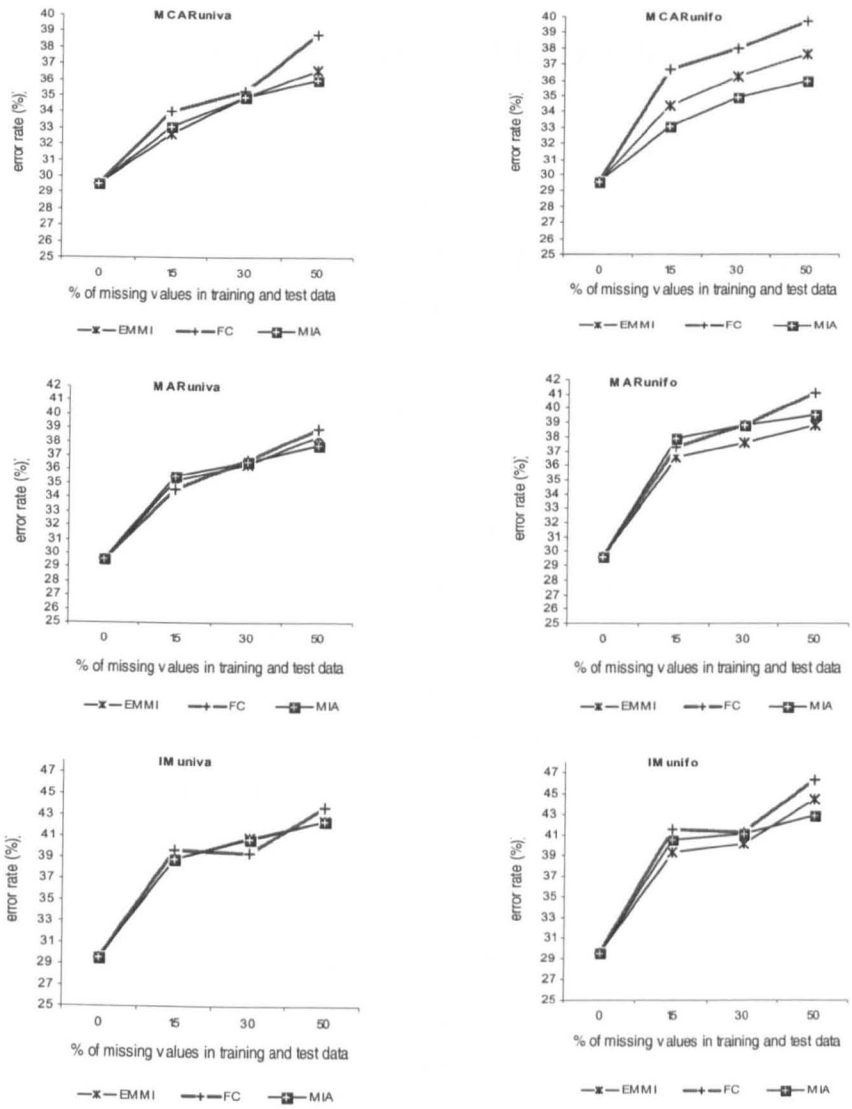

Fig. 5.9. Comparative results of current and proposed methods for the german 24 dataset. A) MCAR*univa*, B) MCAR*unifo*, C) MAR*univa*, D) MAR*unifo*, E) IM*univa*, F) IM*unifo*

EMMI achieves higher accuracy for MAR*unifo* data while MIA's performance appears to improve as the percentage of missing data increases (Figure 5.9D). In this suite of IM*univa* experiments, the results slightly are identical to results observed for MAR*univa* data (Figure 5.9E). In the IM*unifo* case, MIA exhibits another good performance (Figure 5.9F).

5.4.3 Current and New Methods: Processing Time

The two most important issues of Machine Learning algorithms or models are predictive accuracy and speed and scalability. Speed and scalability involves the time spent to construct the model (growing the tree and pruning it) and the time spent to use the model (classifying a new instance using the tree). The training sets are processed until the learning algorithms terminate and then the classification accuracy is measured on the corresponding test sets.

For EMMI, the imputation processing time for EMMI depends on the size of the data matrix and the number of iterations specified for the iterative algorithm. For MIA, three different binary splits which accommodate 'missing' are explored. In fact, the algorithm searches for a good split on all the attributes with missing values, one at a time, and then selects the attribute with the best split. This procedure is carried out individually, i.e., when building the tree and when using the tree to classify an unknown instance.

The processing times (a combination of training and classification times) of the proposed technique against two current methods on three datasets is illustrated. For small datasets, computation time is not a major factor, however if an algorithm needs to be run many times, then it can become an issue. However, the three datasets considered for this exercise are the biggest in the simulation experiments and they encompass purely numerical attributes, purely nominal attributes and mixed attributes, respectively. The idea was to demonstrate experimentally how the proposed technique compares with current methods for handling incomplete training and testing data in terms of computational speed. The only missing data mechanism considered for the exercise is IM (a strong condition) and the 50% proportion of missing values for this condition. Furthermore, the situation when missing values are distributed among all the attributes is considered.

Table 5.4 Processing time (in seconds) for current and proposed methods for selected datasets

Method	Approximate time on a dataset with purely numerical attributes (s)	Approximate time on a dataset with purely nominal attributes (s)	Approximate time on a dataset with mixed attributes (s)	Description
EMMI	31500 (4500)	7100 (1100)	4900 (500)	initial parameter estimates by EM algorithm; iterative simulation; imputation of the missing values
FC	**16300 (2300)**	**4300 (500)**	3900 (400)	exploration of all branches; summing of weights of instance fragments classified in different ways at leaf nodes
MIA	21600 (3600)	5250 (750)	**3300 (300)**	search through all attributes 1. search through split points and send 'missing' to the left and choose the best split; 2. search through split points and send 'missing' to the right and choose the best split; 3. search through split points and send 'missing' to the left and the rest to the right choose the best split of 1 to 3 and best attribute

Table 5.2 displays the approximate running time (given in seconds) for current and proposed methods on datasets with purely numerical attributes (letter); purely categorical attributes (kr-vs-kp); and mixed attributes (german). The processing time given is for both training and testing. Figures in parentheses are for testing only. In addition, the **boldface** font in the table indicates that a method is better than the others.

The quickest method for processing a dataset with purely numerical attributes is FC which is 1.3 times quicker than MIA and about twice as quicker than EMMI. EMMI is also about 1.5 times slower than the proposed technique.

Once again, the results shown in Table 5.2 also show EMMI achieving the slowest running time for handling a dataset with purely nominal attributes, with the proposed technique comparing favourably with FC. In fact the proposed method is not much worse (on average about 1.2 times slower) than FC. However, even though FC is slower than the proposed method it is still about 1.4 times quicker than EMMI.

Results for the dataset with mixed attributes show EMMI achieving the slowest running time, followed by FC (Table 5.2). The quickest running time is by MIA which is about 1.5 times quicker than EMMI and about 1.2 times quicker than FC.

5.5 Discussion

In this chapter a new technique for handling unknown attribute values in both training data and test data has been introduced and its performance compared experimentally against current techniques for handling unknown attribute values on twenty one datasets. The previously proposed methods for handling the incomplete training and test data problems that were used for these experiments are EMMI and FC, while the proposed method is MIA.

The proposed procedure uses three binary splits (whereby each split considers the missingness of the data) in which one chooses the split which maximised the

impurity criterion (the GINI index). The key to making MIA work is the introduction of pseudo value for the missing data. In other words, 'missing' is associated with values of other attributes that are non-missing in the data. This choice of pseudo value allows us to derive an algorithm that can be applied in any kind of dataset and with any kind of attributes.

It appears that the main determinant factor for missing values techniques, especially for smaller percentages of missing values, is the missing data mechanism. However, as the proportion of missing values increases, the distribution of missing values among attributes becomes very important.

The different behaviour of methods according to various missing data mechanisms and missing proportions is remarkable. When examining robustness of missing data techniques for tolerating missing values, FC generally did not perform as well as the other techniques. This is not to say that this technique is without merit, but rather pointing out that there are other techniques that can outperform it. EMMI has the best overall performance. Several statistical theory aspects of EMMI could contribute to explain the apparent advantage conferred by the performance of this method. However, it is interesting to see that for some datasets MIA was more robust to MCAR or MAR data than were EMMI. This is surprising since the one of the assumptions of EMMI algorithm requires that the data be MAR. As expected, MIA consistently performs as well as or significantly better than EMMI for IM data. This conclusion is valid for number of attributes with missing values, different amount of missing values, and missing data mechanisms. As it turns out, there are situations where MIA

One of the strongest arguments against EMMI is that it is much more difficult to implement than some of the other techniques mentioned. One potential problem that was encountered for the EMMI algorithm was convergence. As it turns out, the processing cost of missing values varies for EMMI and MIA with considerable savings in computational time by the latter. EMMI is computationally expensive because it is a greedy search algorithm and can only provide an approximation to the optimal result. The greater the desired accuracy, the greater the computational

cost because of the iterative nature of the algorithm. In fact, for the biggest dataset in the experiment, EMMI took us about one and a half times more to run compared with MIA, even though FC was the quickest of the three methods. Since datasets in the real world are getting bigger by the day, reducing processing cost is one of the most important problems in the machine learning and statistics field. Thus, not only does the proposed technique deal with missing values as well as EMMI it handles incomplete training and test data at a lower cost.

MIA is suitable for a wide range of datasets as it does not make representational assumptions or presupposes other model constraints.

The datasets used in our experiment have also provided a valuable insight into the limitations of the missing data techniques. Each dataset appears more or less to have its own 'favourite' technique for handling unknown attribute values. However, this depends on random variation, the magnitude of missing values and the source of missingness in each dataset.

It has also been found that missing values have more impact when data is informatively missing than when it is missing at random or missing completely at random, i.e., IM values entail deterioration in predictive accuracy compared with MCAR and MAR. This is somehow in accordance with missing data theory, which suggests that non-randomly missing parameters are subject to bias. In other words, IM data is difficult to deal with. However, the proposed method appears to handle IM data slightly better compared with the other methods.

Chapter 6

More on the Problem of Classifying Incomplete Vectors Using Trees

6.1 Introduction

One of the central tasks of supervised learning algorithms is classifying instances from some domain of application, i.e., determining whether a particular instance belongs to a specified class, given a description of that instance. Virtually all research on supervised learning addresses the task of learning to classify complete domain instances. However, in many real world situations we often have to classify instances given incomplete vectors.

The task of learning an accurate incomplete data classifier from instances raises a number of new issues which have not been addressed by traditional supervised learning research. First, the types of processes that can cause an instance to have missing attribute values have to be considered. For example, whether this omission is randomly missing, uninformative, partially informative, or even misleading. Second, training on incomplete data versus training on the artificially completed instances can also be considered. Intuitively, complete data give the learner more information about each instance, and hence, should make learning easier.

Although the task of learning DTs for incomplete data has been considered in a few empirical studies, the results of these studies have invariably been mixed: a number of techniques for handling incomplete test data have been investigated, but none has been found to be uniformly superior to the others. Part of the problem is that this learning task has yet to receive the same degree of theoretical treatment as learning from complete data. So, there is no explanation of this phenomenon. However, the

results of the experiments carried out in chapter 4 gives an indication of which technique is best under specific circumstances.

The results and experience obtained in the chapter 4 suggested to us to come up with simple but efficient solutions for handling unknown (undetermined) attribute values in the test set. According to results of the previous experiments, missing values appeared to have more impact when they occur in the test set. Also, probabilistic methods showed very good results. In fact, it is easy to say that methods like EMMI and FC can have high performance in missing values problems but are open to improvements. Hence, algorithms based on a probabilistic approach for handling incomplete test data were of significant practical interest.

The purpose of this chapter is to develop probabilistic methods for classifying incomplete vectors using decision trees, i.e. methods that could be used to handle incomplete test data. This approach is based on the *a priori* probability of each value determined from the instances at that node that have specified values. The missing attribute values can be either continuous or nominal. The proposed method follows the total probability and Bayes' theorems (Press, 1989; Bernardo and Smith, 1994) and it has three versions. The behaviour of the proposed approach is explored by a simulation study. The effects of five methods of handling missing attribute values in test (unseen) data are experimentally investigated by comparing two of the best current methods (from the results of our previous experiments) with the three proposed probabilistic methods from the point of view of their effects or tolerance of incomplete test data.

With this approach, a test instance being classified is passed down all the possible branches of the tree corresponding to the value of the attribute. This process is repeated at each node on this branch and so on until a leaf node is reached. After the new instance is passed down to all the possible leaves, the class probabilities of these leaves are combined and finally the test instance is assigned with the biggest probability. Thus, the main idea of the proposed approach is to estimate the probabilities that are used for predicting membership of a particular class given that

there are unknown or missing attribute values in test data. These probabilities are estimated with respect to each class as given below:

First, these probabilities were estimated by using instances from the training set (TSPE). Secondly, the probabilities were predicted using the instances from the decision tree (DTPE). Finally, an attempt was made to improve the prediction of each class membership, thus classification accuracy, by applying some type of 'smoothing' procedure to estimate the conditional probabilities involving the class variable and the other attribute variables. The procedure was carried out utilising either the binary or multinomial logit models, depending on the class variable distribution of that particular dataset. The binary logit model was used for the datasets with only two classes while the multinomial logit model was used for those datasets with more than two classes for the response variable. This approach shall now be called logistic probability estimation (LPE)

The remainder of this chapter is organized as follows. In the next section, the framework of the proposed probabilistic method is introduced and described. Experiments exhibiting the performance of the proposed method and existing methods are presented in Section 6.3, and the results are analyzed in Section 6.4. The chapter concludes with a brief discussion.

6.2 Classifying Incomplete Vectors Using Trees – Proposed Procedure

The proposed probabilistic approach to missing attribute values follows both branches from each node if the value of the attribute being branched on is not known.

Given n mutually exclusive events $X_1,..., X_n$ whose probabilities sum to unity, then

$$P(Y) = \sum_{i=1}^{n} P(Y \mid X_i)P(X_i), \tag{6.1}$$

where Y is an arbitrary event, and $P(Y \mid X_i)$ is the conditional probability of Y assuming X_i. This is the theorem of total probability.

The total probability theorem and the definition of conditional probability (introducing an arbitrary event Z) may be used to derive

$$P(Y \mid Z) = \sum_{i=1}^{n} P(Y \mid Z, X_i) P(X_i \mid Z) \tag{6.2}$$

The missing value problem addressed in this thesis can be defined as follows:

Given: A decision tree, a complete set of training data, and a set of instances for testing, described with attributes and their values. Some of the attribute values in the test instances are unknown.

Find: A classification rule for a new instance using the tree structure given that it has an unknown attribute value and by using the known attribute values.

Let A be the attribute associated with a particular node of the tree that could either be discrete or numerical. A discrete attribute has a certain number of possible values J and a continuous attribute may attain any value from a continuous interval. Each node is split into two sons (left and right sons). Hence, a new instance could either go to the left (L) or to the right (R) of each internal node. Further, let V be the binarised value for attribute A. Let C denote a class and let there be k classes, $j = 1,...,k$.

The total probability theorem is used to predict class membership of an unknown attribute value by computing the conditional probability of a class C given the evidence of known attribute values.

For individual j, divide the attributes in the tree into classes for both $\underline{K}$ (the known attribute values) and $\underline{M}$ (the missing attribute values). Then

$$P(C_j \mid \underline{K}) = \sum P(C_j \mid \underline{K}, \underline{M}) P(\underline{M}) \tag{6.3}$$

where the sum is over all possible combinations of values that branch to the left (L) or right (R) at each respective internal node, taken by the vector of the missing attribute values $\underline{M}$. For the unknown attribute values, the unit probability may be distributed across the various leaves to which the new instance could belong. These

probabilities are going to be estimated in three ways as explained in the following example.

For illustration purposes, suppose that from Figure 6.1 the values for A_1 (categorical attribute) and A_3 (numerical attribute) are missing, and A_2 is the only numeric attribute with non-missing values. The tree is constructed using artificial data given in Table 6.1. This dataset is identical to the one used for the illustrative example in Chapter 5 but it has no missing attribute values.

From the example, it appears from Figure 6.1 that all the attributes with no missing values would be used when estimating the probabilities. However, this does not have to be the case. The attributes that are used are determined by where the instance branches at a particular internal node. For example, say, attribute A_1 was not missing. For any instance branching to the left of A_1 that would mean non-utilisation of attribute A_2, which is connected to the right branch of A_1.

First Case: Class membership for a new instance is predicted given that it will branch to the left of the internal node A_2 (A_2^L), given that both A_1 and A_3 have unknown attribute values.

The probability that the predicted class membership will be class 1 given that it branches to the left at internal attribute 2 ($P(C_1 \mid A_2^L)$) can be defined by:

$$\begin{aligned} P(C_1 \mid A_2^L) = {} & P(C_1 \mid A_2^L, A_1^L, A_3^L)P(A_1^L, A_3^L \mid A_2^L) \\ & + P(C_1 \mid A_2^L, A_1^L, A_3^R)P(A_1^L, A_3^R \mid A_2^L) \\ & + P(C_1 \mid A_2^L, A_1^R, A_3^L)P(A_1^R, A_3^L \mid A_2^L) \\ & + P(C_1 \mid A_2^L, A_1^R, A_3^R)P(A_1^R, A_3^R \mid A_2^L) \end{aligned} \tag{6.4}$$

Similarly,

$$\begin{aligned} P(C_2 \mid A_2^L) = {} & P(C_2 \mid A_2^L, A_1^L, A_3^L)P(A_1^L, A_3^L \mid A_2^L) \\ & + P(C_2 \mid A_2^L, A_1^L, A_3^R)P(A_1^L, A_3^R \mid A_2^L) \\ & + P(C_2 \mid A_2^L, A_1^R, A_3^L)P(A_1^R, A_3^L \mid A_2^L) \\ & + P(C_2 \mid A_2^L, A_1^R, A_3^R)P(A_1^R, A_3^R \mid A_2^L) \\ & = 1 - P(C_1 \mid A_2^L) \end{aligned} \tag{6.5}$$

Table 6.1 Artificial dataset

A_1	A_2	A_3	*Class*
4	936	9	1
1	1168	12	1
4	5117	27	1
1	902	12	2
4	1495	12	1
1	10623	30	1
4	1935	12	1
2	1424	12	1
1	6568	24	1
4	1413	12	1
4	3074	9	1
4	3835	36	1
1	5293	27	2
3	1908	30	2
4	3342	36	1
2	932	6	1
1	3104	18	1
3	3913	36	1
1	3021	24	1
4	1364	10	1
2	625	12	1
1	1200	12	1
4	707	12	1
4	2978	24	1
4	4657	15	1
4	2613	36	1
2	10961	48	2
1	7865	12	2
4	1478	9	2
1	3149	24	1
3	4210	36	2
4	2507	9	1
4	2141	12	1
2	866	18	1
4	1544	4	1
1	1823	24	2
2	14555	6	2
2	2767	21	2
4	1291	12	1
1	2522	30	1

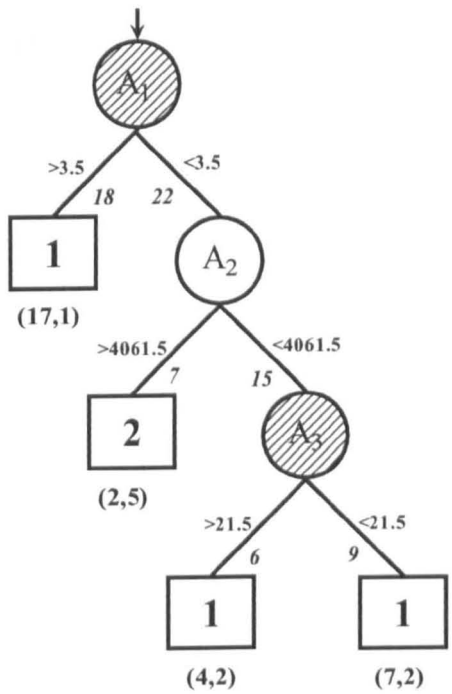

Figure 6:1 Example of a binary decision tree from a set of 40 training instances that are represented by three attributes and accompanied by two classes

Note: Figures in brackets are the number of instances in each terminal node for class 1 and 2, respectively. Figures in italic represent training data instances that branch to the right or the left of each internal node at each respective cut-off point.

We will use the following two methods of estimating the above probabilities.

6.2.1 The Full Estimation of Probabilities from Training Set

From Table 6.1, there are 1 class 1 individual associated with A_2^L ($A_2 > 4061.5$), A_1^L ($A_1 > 3.5$), A_3^L ($A_3 > 21.5$), 1 class 1 individual with A_1^L, A_2^L, A_3^L and so on. In addition one of the 7 A_2^L individuals (with $A_2 > 4061.5$) has A_1^L, A_3^L, another 1 has A_1^L, A_3^R, and so on. Therefore, the estimated probability of membership of class 1 is given by:

$$
\begin{aligned}
P(C_1 \mid A_2^L) &= \left(\frac{\left(\frac{1}{40}\right)}{\left(\frac{1}{40}\right)}\right)\left(\frac{\left(\frac{1}{40}\right)}{\left(\frac{9}{40}\right)}\right) + \left(\frac{\left(\frac{1}{40}\right)}{\left(\frac{1}{40}\right)}\right)\left(\frac{\left(\frac{1}{40}\right)}{\left(\frac{9}{40}\right)}\right) + \left(\frac{\left(\frac{2}{40}\right)}{\left(\frac{5}{40}\right)}\right)\left(\frac{\left(\frac{5}{40}\right)}{\left(\frac{9}{40}\right)}\right) + \left(\frac{\left(\frac{0}{40}\right)}{\left(\frac{2}{40}\right)}\right)\left(\frac{\left(\frac{2}{40}\right)}{\left(\frac{9}{40}\right)}\right) \\
&= \left(\frac{1}{9}\right) + \left(\frac{1}{9}\right) + \left(\frac{2}{9}\right) \\
&= \left(\frac{4}{9}\right) \\
&= 0.444.
\end{aligned}
\tag{6.6}
$$

Following from (6.6), $P(C_2 \mid A_2^L) = 1 - P(C_1 \mid A_2^L) = 0.556$ where $P(C_1 \mid A_2^L)$ and $P(C_2 \mid A_2^L)$ are both estimated from the proportion of instances in the training set for which this is true, respectively. The assumption is that there is "exchangeability" between all the instances in the training and test data.

For this method the unknown instance will be classified as belonging to class 2 as it has the highest probability.

From Table 6.1, the estimated probability of membership of class 1 is given by:

$$
\begin{aligned}
P(C_1 \mid A_2^R) &= \left(\frac{\left(\frac{4}{40}\right)}{\left(\frac{4}{40}\right)}\right)\left(\frac{\left(\frac{4}{40}\right)}{\left(\frac{31}{40}\right)}\right)+\left(\frac{\left(\frac{11}{40}\right)}{\left(\frac{12}{40}\right)}\right)\left(\frac{\left(\frac{12}{40}\right)}{\left(\frac{31}{40}\right)}\right)+\left(\frac{\left(\frac{4}{40}\right)}{\left(\frac{6}{40}\right)}\right)\left(\frac{\left(\frac{6}{40}\right)}{\left(\frac{31}{40}\right)}\right)+\left(\frac{\left(\frac{7}{40}\right)}{\left(\frac{9}{40}\right)}\right)\left(\frac{\left(\frac{9}{40}\right)}{\left(\frac{31}{40}\right)}\right) \\
&= \left(\frac{4}{31}\right)+\left(\frac{11}{31}\right)+\left(\frac{4}{31}\right)+\left(\frac{7}{31}\right) \\
&= \left(\frac{26}{31}\right) \\
&= 0.839.
\end{aligned} \tag{6.7}
$$

Following from (6.6), $P(C_2 \mid A_2^R) = 1 - P(C_1 \mid A_2^R) = 0.161$ where $P(C_1 \mid A_2^R)$ and $P(C_2 \mid A_2^R)$ are both estimated from the proportion of instances in the training set for which this is true, respectively.

6.2.2 Approximation of Probabilities by Related Probabilities Estimated from Decision Tree

We shall now approximate the same probabilities for each respective class given the known attribute values by using the instances given from the decision tree only (Fig. 6.1) instead of using the training data instances used for the first method. These probabilities are used to predict class membership for an unknown test instance.

From formula (6.4), the probabilities can be approximated using the decision tree shown in Figure 6.1 by:

$$
\begin{aligned}
P(C_1 \mid A_2^L) &\approx P(C_1 \mid A_2^L, A_1^L)P(A_1^L \mid A_2^L) + P(C_1 \mid A_2^L, A_1^R)P(A_1^R \mid A_2^L) \\
&\approx P(C_1 \mid A_1^L)P(A_1^L) + P(C_1 \mid A_2^L, A_1^R)P(A_1^R) \\
&= \left(\frac{17}{18}\right)\left(\frac{18}{40}\right)+\left(\frac{2}{7}\right)\left(\frac{22}{40}\right) \\
&= \frac{163}{280} \\
&= 0.582.
\end{aligned} \tag{6.8}
$$

$$P(C_2 \mid A_2^L) = 1 - P(C_1 \mid A_2^L) \\ = 0.418. \tag{6.9}$$

Hence, the unknown instance will be classified as belonging to class 1.

The probabilities $P(C_1 \mid A_2^L, A_1^L)$ and $P(C_1 \mid A_2^L, A_1^R)$ are estimated by $P(C_1 \mid A_1^L)$ and $P(C_1 \mid A_1^R)$, respectively. Also, $P(A_1^L \mid A_2^L)$ and $P(A_1^R \mid A_2^L)$ approximated by $P(A_1^L)$ and $P(A_1^R)$ respectively. $P(A_1^R)$ is the probability that the new instance will branch to the right of attribute A_1. To summarise, it could be said that conditioning is not taking place on A_2^L.

It is easy to notice from formula (6.7) that any conditional probability involving A_3 has not been considered for this particular method. This is because whenever an instance branches to the left of A_2, A_3 would automatically not take part in the classification stage, hence, being left out of all the calculations.

Second Case: Class membership for a new instance given that it will branch to the right of the internal node A_2 (A_2^R) is predicted, given that both A_1 and A_3 have unknown attribute values. The class with the biggest probability is selected.

We can define the probability that the predicted class membership will be class 1 given that it branches to the right of internal attribute 2 ($P(C_1 \mid A_2^R)$) by:

$$\begin{aligned} P(C_1 \mid A_2^R) = {} & P(C_1 \mid A_2^R, A_1^L, A_3^L) P(A_1^L, A_3^L \mid A_2^R) \\ & + P(C_1 \mid A_2^R, A_1^L, A_3^R) P(A_1^L, A_3^R \mid A_2^R) \\ & + P(C_1 \mid A_2^R, A_1^R, A_3^L) P(A_1^R, A_3^L \mid A_2^R) \\ & + P(C_1 \mid A_2^R, A_1^R, A_3^R) P(A_1^R, A_3^R \mid A_2^R) \end{aligned} \tag{6.10}$$

Similarly,

$$
\begin{aligned}
P(C_2 \mid A_2^R) = {} & P(C_2 \mid A_2^R, A_1^L, A_3^L)P(A_1^L, A_3^L \mid A_2^R) \\
& + P(C_2 \mid A_2^R, A_1^L, A_3^R)P(A_1^L, A_3^R \mid A_2^R) \\
& + P(C_2 \mid A_2^R, A_1^R, A_3^L)P(A_1^R, A_3^L \mid A_2^R) \\
& + P(C_2 \mid A_2^R, A_1^R, A_3^R)P(A_1^R, A_3^R \mid A_2^R) \\
& = 1 - P(C_1 \mid A_2^R)
\end{aligned}
\tag{6.11}
$$

We will use the following two methods of estimating the above probabilities.

From Figure 6.1, the estimated probability of membership of class 1 is given by:

$$
P(C_1 \mid A_2^R) = P(C_1 \mid A_2^R, A_1^R)P(A_1^R \mid A_2^R) + P(C_1 \mid A_2^R, A_1^L)P(A_1^L \mid A_2^R) \tag{6.12}
$$

where
$$
\begin{aligned}
P(C_1 \mid A_2^R, A_1^R) = {} & P(C_1 \mid A_2^R, A_1^R, A_3^L)P(A_3^L \mid A_1^R, A_2^R) \\
& + P(C_1 \mid A_2^R, A_1^R, A_3^R)P(A_3^L \mid A_1^R, A_2^R) \\
= {} & \left(\frac{4}{6}\right)\left(\frac{6}{15}\right) + \left(\frac{7}{9}\right)\left(\frac{9}{15}\right) \\
= {} & \frac{11}{15}
\end{aligned}
$$

Therefore, $P(C_1 \mid A_2^R) \approx \left(\frac{11}{15}\right)\left(\frac{22}{40}\right) + \left(\frac{17}{18}\right)\left(\frac{18}{40}\right) = \frac{497}{600} = 0.828.$

Using (6.12),

$$
\begin{aligned}
P(C_2 \mid A_2^R) &= P(C_2 \mid A_2^R, A_1^R)P(A_1^R \mid A_2^R) + P(C_2 \mid A_2^R, A_1^L)P(A_1^L \mid A_2^R) \\
&= 1 - P(C_1 \mid A_2^R) \\
&= 0.172.
\end{aligned}
\tag{6.13}
$$

As with the first case, the probabilities $P(C_1 \mid A_2^L, A_1^L)$ and $P(C_1 \mid A_2^R, A_1^L)$ are estimated by $P(C_1 \mid A_1^L)$ and $P(C_1 \mid A_1^R)$, respectively. Also, $P(A_1^L \mid A_2^R)$, $P(A_1^R \mid A_2^R)$ $P(A_1^R, A_3^L \mid A_2^R)$ and $P(A_1^R, A_3^R \mid A_2^R)$ approximated from the tree shown in Figure 6.1 by $P(A_1^L)$, $P(A_1^R)$, $P(A_3^L)$ and $P(A_3^R)$ respectively. To summarise, it could be said that conditioning is not taking place on A_2^R.

In the first case, where a given new instance will branch to the left of the internal node given that there are unknown attribute values, class 2 will be selected by the first method and class 1 selected by the second method. They both have the highest probabilities of 0.556 and 0.585, respectively. For the second case, where a given new instance will branch to the right of the internal node given that there are unknown attribute values, class 1 will be selected for both methods, which have much more clear-cut 'bigger' probabilities of 0.839 and 0.828, respectively.

6.2.3 The Full Estimation of Probabilities from Training Set Using Binary and Multinomial Logit Models

In this sub-section the estimation of probabilities for the proposed probabilistic method are improved by using logistic regression (Agresti, 1990; McCullagh and Nelder, 1990; Collett, 1991) and multinomial logit techniques (Hosmer *et al.*, 1989; Agresti, 1990; Long, 1997), individually. The binary logit model (BLM) is used to estimate probabilities for those datasets that have two classes with the latter used to estimate probabilities for datasets with more than two classes.

For example, suppose that there are two classes, 1 and 2, (C_1 and C_2) and v attribute variables $A_1,...,A_v$. Then the probability that an object with values $a_1,...,a_v$ belongs to class 1 as a logistic function of the $A_1,...,A_v$ could be modelled:

$$P(C_1 \mid A) = \frac{\exp\{\beta_0 + \beta_1 A_1 + ... + \beta_k A_k\}}{1 + \exp\{\beta_0 + \beta_1 A_1 + ... + \beta_k A_k\}} \qquad (6.14)$$

and then estimate the unknown parameters β_i from the training data on objects with known classifications.

BLMs, like logistic regression, describe the relationship between a dichotomous response variable and a set of explanatory variables of any type. The explanatory variables may be continuous or categorical variables. Binary logit tries to model the

logarithmic odds-ratio for the classification (dependent variable C) as a linear function of the v 'input' or attribute variables $\vec{A} = A_1,..., A_v$.

For purposes of this thesis, the BLM was not used to estimate probabilities based on all the attributes given in the dataset, but to estimate only the unknown probabilities of the given attributes specifically related to the problem. For each specific attribute, the values of the instances were made binary in accordance to the branching of that particular value at the internal node of the tree, i.e., whether the value branched to the left or to the right at the internal node. For example, if the value branched to the left of the internal node of interest, it was recorded as 1. Otherwise, it was recorded as 2.

For the two-class example discussed in Section 6.1, the conditional probabilities involving only the class given in equations 6.4 and 6.5 ($P(C_1 \mid A_2^L, A_1^L, A_3^L)$, $P(C_1 \mid A_2^L, A_1^L, A_3^R)$, $P(C_1 \mid A_2^L, A_1^R, A_3^L)$ and $P(C_1 \mid A_2^L, A_1^R, A_3^R)$, for class 1, and $P(C_2 \mid A_2^L, A_1^L, A_3^L)$, $P(C_2 \mid A_2^L, A_1^L, A_3^R)$, $P(C_2 \mid A_2^L, A_1^R, A_3^L)$ and $P(C_2 \mid A_2^L, A_1^R, A_3^R)$, for class 2), could be estimated by the binary logit model in terms of the log odds ratio in the form:

$$\log\left[\frac{P(C_1 \mid \vec{A})}{P(C_2 \mid \vec{A})}\right] = \beta_0 + \beta_1 A_1 + ... + \beta_k A_k = \beta_0 + \vec{\beta}\,\vec{A}^T \qquad (6.15)$$

where $\vec{\beta}$ is the k dimensional coefficient vector. The odds ratio is a factor of how many times the event (C_1) is more likely to happen than event (C_2), given the knowledge of A.

For an example $P(C_1 \mid A_2^L, A_1^L, A_3^L)$ is estimated by:

$$\log\left[\frac{P(C_1 \mid A_2^L, A_1^L, A_3^L)}{P(C_2 \mid A_2^L, A_1^L, A_3^L)}\right] = \beta_0 + \beta_1 A_2^L + \beta_2 A_2^L + \beta_3 A_3^L$$

Although binary logit model finds the best 'fitting' equation just as the linear regression does, the principles on which it does so are different. Instead of using the least-squares deviations criterion for the best fit, it uses a maximum likelihood method, which maximises the probability of getting the observed results given the fitted regression coefficients.

A model that could be used for a dependent variable that has only two possible categories or two classes for the example has been discussed. However, logistic regression could be extended to accommodate an analysis of dependent variables that have more than two possible categories, which could either be ordered or unordered. In other words, an approach that would be able to handle a problem with three or more classes. This type of logistic regression approach is known as the multinomial logit model (MLM) and has the following form: for *k+1* classes:

$$P(C_j) = \frac{\exp(\vec{\beta}_j^T \vec{X})}{\sum_{j=1}^{k+1} \exp(\vec{\beta}_j^T \vec{X})} \qquad \text{for } j = 1,\ldots, k+1 \tag{6.16}$$

which will automatically yield probabilities that sum to unity.

In order to identify the parameters of the model, β_{k+1} is set to 0 (a zero vector) as a normalisation procedure and thus:

$$P(C_{k+1}) = \frac{1}{\sum_{j=1}^{k+1} \exp(\vec{\beta}_j^T \vec{X})}. \tag{6.17}$$

In the MLM model the assumption is that the log-odds of each response follow a linear model. Thus, the j^{th} logit has the following form:

$$\log\left[\frac{P(C_j)}{P(C_{k+1})}\right] = \vec{\beta}_j^T \vec{X} \tag{6.18}$$

where β_j is a vector of regression coefficients for $j = 1,\ldots, k$. This model is analogous to the LR model, except that the probability distribution of the response is multinomial instead of binomial and there are k equations instead of one. The k

multinomial logit equations contrast each of categories $j = 1,...,k$ with category $k+1$, whereas a single logistic regression equation is a contrast between successes and failures. If $k = 1$ the multinomial logit model reduces to the usual binary logit model. The multinomial logit model is in fact equivalent to running a series of BLMs.

The reader is referred to Section 2.1.1.3 for the calculation of probabilities.

Once again, for more details about the MLM and how the logits and probabilities are modelled, the reader is referred to (Hosmer *et al.*, 1989; Agresti, 1990; Long, 1997).

For all datasets with more than two classes, the conditional probabilities used to predict a class for a new instance were modelled using the MLM.

There are similarities between the proposed method and FC (Quinlan, 1993). For example, when classifying a new instance with an unknown value for the attribute being tested, all branches are explored and the results are combined to reflect relative probabilities of different outcomes. Also, the processing time of both methods grows exponentially as a function of the number of missing feature values and becomes intractable for large datasets with large numbers of missing values.

However, there are differences between the proposed method and FC on how these probabilities are estimated. FC "fraction' instances based on the a priori probability of each value determined from the instances at that node that have specified values. A test instance is fractioned according to the training instances at nodes that test features which it is missing. For example, given a Boolean attribute A_2, if node n contains 7 known instances with A_2^L and 15 with A_2^R, then the probability that A_2^L = 0.318, and the probability that A_2^R =0.682. A fraction 0.318 of instance x is now distributed down the branch for A_2^L and a fractional 0.682 of x down the other branch. Each fraction is classified down a leaf (possible being fractioned again) and then the total fraction of the instance assigned to each category summed and the instance is finally assigned to the category with the overall largest fraction. In other words, the classification of the new instance is simply the most probable classification, computed by following all branches at any node for which a value is

missing, multiplying and summing the weights of the instance fragments classified in different ways at the leaf nodes of the tree.

For example, using Figure 6.1 (given that A_1 and A_3 are missing, i.e., [??, A_2, ??]). The probability of membership of class 1 is given by:

$$P(C_1 \mid ??, A_2, ??) = P(C_1 \mid ??, A_2^R, ??)$$
$$= \frac{18}{40} + \left(\frac{22}{40}\right)\left(\frac{15}{22}\right)\left(\frac{6}{15}\right) + \left(\frac{22}{40}\right)\left(\frac{15}{22}\right)\left(\frac{9}{15}\right)$$
$$= \frac{33}{40}$$

and

$$P(C_2 \mid ??, A_2, ??) = P(C_2 \mid ??, A_2^L, ??)$$
$$= \frac{7}{40} \quad [\text{or } 1 - P(C_1 \mid ??, A_2^R, ??)]$$

Hence, the new instance would finally be classified as class 1.

Notice that whereas the proposed procedure considers only those instances belonging to that particular class for which an unknown instance would be classified, FC considers all the instances branching to that particular leaf node whose class is being predicted, and which would be given at the particular leaf node. For example, when classifying that an unknown instance would be class 2, at the internal node A_2^L, FC takes all the 7 instances branching to the leaf node in its probability distribution while for the proposed approach only the 5 instances related to class 2 in the distributions are considered.

6.3 Experimental Set-Up

In this section the behaviour of the three proposed procedures against two approaches that have previously been proposed for handling unknown attribute values in test data when using decision trees are explored. The two current methods selected (EMMI and FC) are the ones which provided very good results in the experiments carried out in Chapter 4. Since the main objective is to compare the performance of the proposed method with current approaches to deal with the

problem of incomplete test data. The performances of TSPE, DTPE and LPE, on the one hand, against FC and EMMI, on the other hand, are experimentally compared. Once again, EMMI is used as a baseline as it was clearly 'the winner' in previous experiments in Chapter 4. In addition, since the proposed algorithm is superficially similar to FC (one of the most well known machine learning algorithm), it was of important to explore how accurate it is relative to FC.

The five methods were run at different proportions of missingness using different missing data mechanisms. All combinations were tested on all twenty one datasets and executed five times for each combination. The experiment is similar to that described in previous experimental sections. Hence, detailed experimental methods are not included but only a subset of the experiment.

6.4 Experimental Results

Experimental results on the effects of current and proposed methods for handling incomplete test data on predictive accuracy using trees are described. The behaviour of these methods is explored for two conditions determining the number of attributes with missing value, four different levels of missing values in test data, and for the MCAR, MAR and IM patterns of missing values. The results are presented in two parts. The first section compares the performance of five different approaches for classifying incomplete vectors using decision trees, looking at the overall results of each method, averaged over all twenty one datasets. The second section compares the performance of the five methods on individual datasets, especially the datasets where some interesting trends achieved by the proposed and current methods emerged.

6.4.1 Overall Results - Current Vs. New Testing Methods

Figure 6.2 summarises the overall excess error rates for current and new testing methods against three amounts of missing values. The error rates of each method of dealing with the introduced missing values are averaged over the 21 datasets.

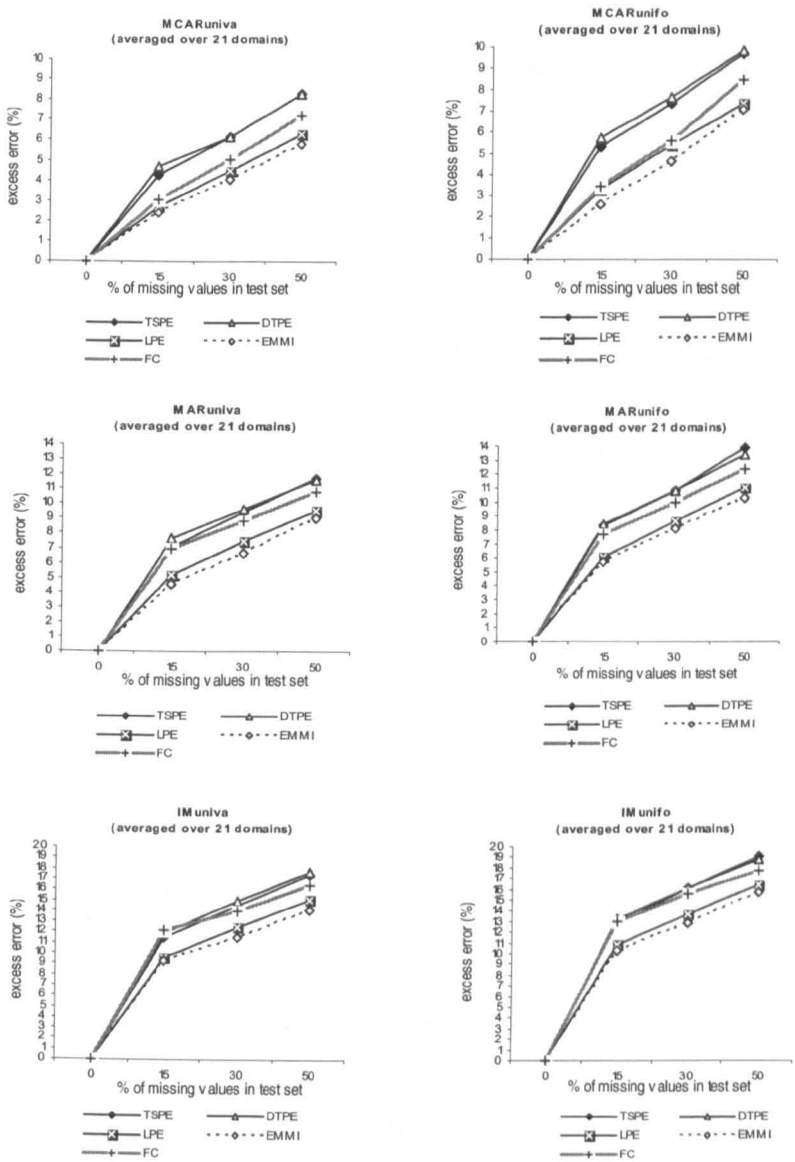

Fig. 6.2. Effects of missing values in test data on excess error for current and proposed testing methods. A) MCAR*univa*, B) MCAR*unifo*, C) MAR*univa*, D) MAR*unifo*, E) IM*univa*, F) IM*unifo*

From Figure 6.2A, both EMMI and LPE are more robust to MCAR*univa* data while TSPE shows more deterioration in performance with increasing amount of missing data. Figure 6.2B presents error rates of methods for MCAR*unifo* data which are similar to results for the MCAR*univa* suite. The results in Figure 6.3C show TSPE as more effective as a method for handling MAR*univa* data than MCAR*univa* data. Results for MAR*unifo* data, shows a similar pattern of results to the one observed for MCAR*unifo* data (Figure 6.2D). The results in Figure 6.2E show poor performances

by TSPE and DTPE for IM*univa* data. It can be seen from Figure 6.2F that results yielded by methods for IM*unifo* data are identical to results achieved by methods for MAR*unifo* data.

It seems that the overall performance of LPE is rather effective on average compared with TSPE and DTPE, and also gives EMMI serious competition. This is the case for all the three missing data mechanisms. The slightly better performance of DTPE compared with TSPE in some situations, especially at higher levels of missing values, is rather surprising. This is considering the fact that for this technique the probabilities are not estimated in the correct way but by using the information given on the tree.

Main Effects

To determine how many of the main individual factors and the respective interactions are significant, the ANOVA was carried out. The experimental results given in Table 6.2 in the Appendix suggest that the existing and new methods for handling incomplete test data, the proportion of incomplete test data and the missing data mechanisms are statistically significant in classification performance at the 1% level.

The performance of the missing data methods is summarised in Figure 6.3. The best method for handling incomplete test data using decision trees is EMMI, followed by LPE, FC, TSPE and DTPE, respectively. There also appears to be small differences in error rate between TSPE and DTPE, on the one hand, and LPE and EMMI, on the other hand (Figure 6.3).

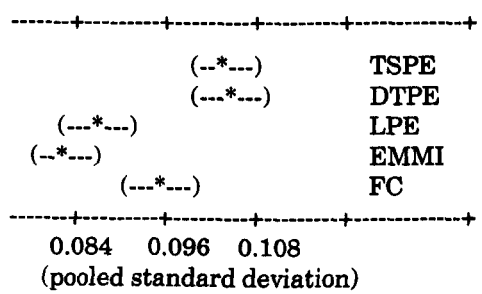

Fig. 6.3. Comparison for current and proposed testing methods: confidence intervals of mean error rates (*)

Notice that the relative effects of number of attributes with missing values, proportion of missing values and the effect of different missing data mechanisms are just as they were for the different collection of methods in Chapters 4 and 5.

Interaction Effects

Three two-way and one three-way interactions were found to be statistically significant at the 1 % level (See Table 6.2 in the Appendix). Only two of the two-way interactions that were significant for current testing methods in previous experiments are also significant for this experiment. The significant three way interaction is between testing methods, missing data proportions and missing data mechanisms. The results of the experiment are once again in accord with previous experiments in chapters 4 and 5 and they are not discussed (to avoid repetition between chapters).

6.4.2 Results for Individual Datasets - Current Vs. New Testing Methods

As it was the case in previous chapters, the results that illustrate specific deviations from the overall results of the effectiveness of the proposed method against the current methods for classifying incomplete vectors on different database characteristics, especially on datasets with purely nominal attributes and mixed attributes are presented. The results for the letter dataset problem (with purely numerical attributes) follow a similar pattern to results obtained for the same dataset in chapter 5 and are not covered in this section.

6.4.2.1 Results on a Dataset with Purely Nominal Attributes: kr-vs-kp

For the kr-vs-kp data problem, the effects of missing values on classification accuracy for MAR*univa* data are summarised in Figure 6.4A. Good performances are observed for EMMI and LPE. The results achieved by methods for MCAR*unifo* data are identical to results obtained for MCAR*univa* data (Figure 6.4B). For MAR*univa* data, FC's performance improves with increasing amount of missing data

(Figure 6.4C). For MAR*unifo* data, both EMMI and LPE are as effective as shown in Figure 6.4D. The impact of IM*univa* data on classification accuracy is shown in Figure 6.4E with DTPE achieving the worst performance (Figure 6.4E). In this suite of IM*unifo* experiments, the behaviour of the methods shown in Figure 6.4F is not different from the one observed in the MAR*unifo* case.

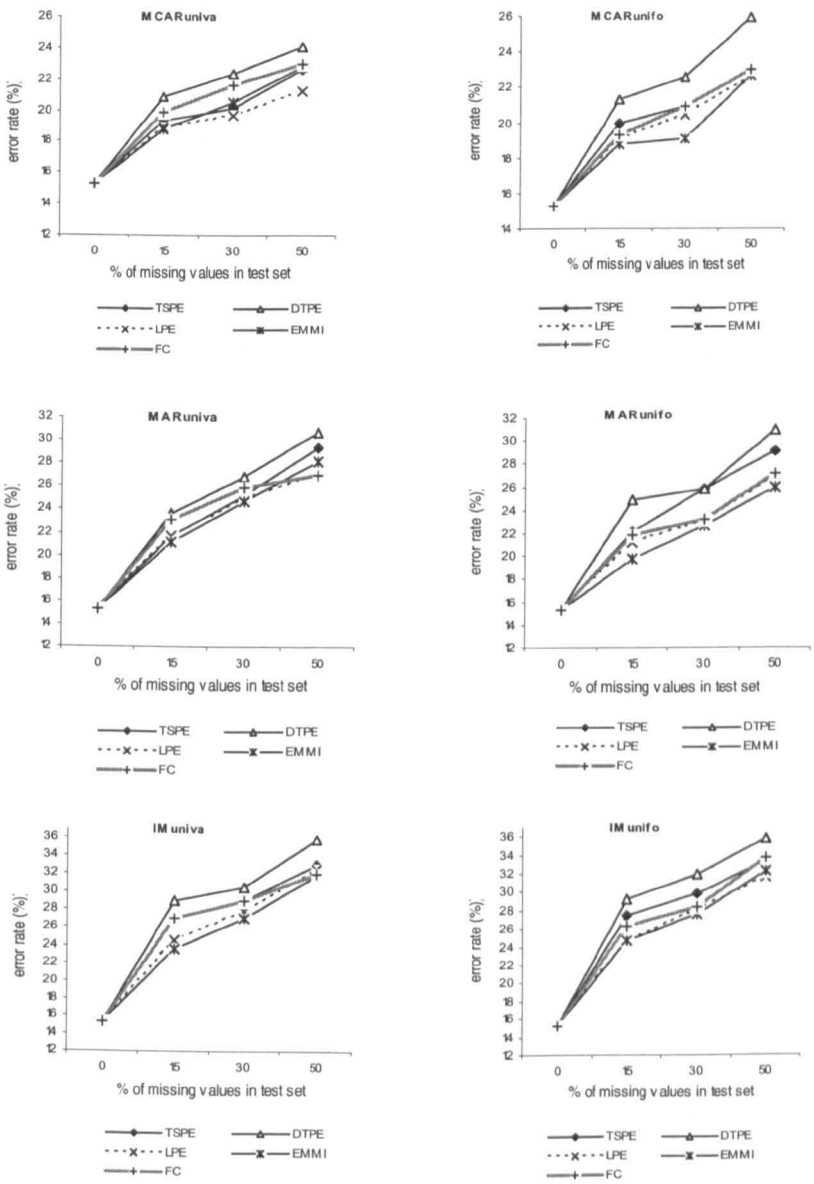

Fig. 6.4. Comparative results of current and proposed testing methods for the kr-vs-kp data. A) MCAR*univa*, B) MCAR*unifo*, C) MAR*univa*, D) MAR*unifo*, E) IM*univa*, F) IM*unifo*

6.4.2.2 Results on Dataset with Mixed Attributes: german

From Figure 6.5A, the overall best performance for MCAR*univa* data is by LPE while biggest error rates are achieved by TSPE.

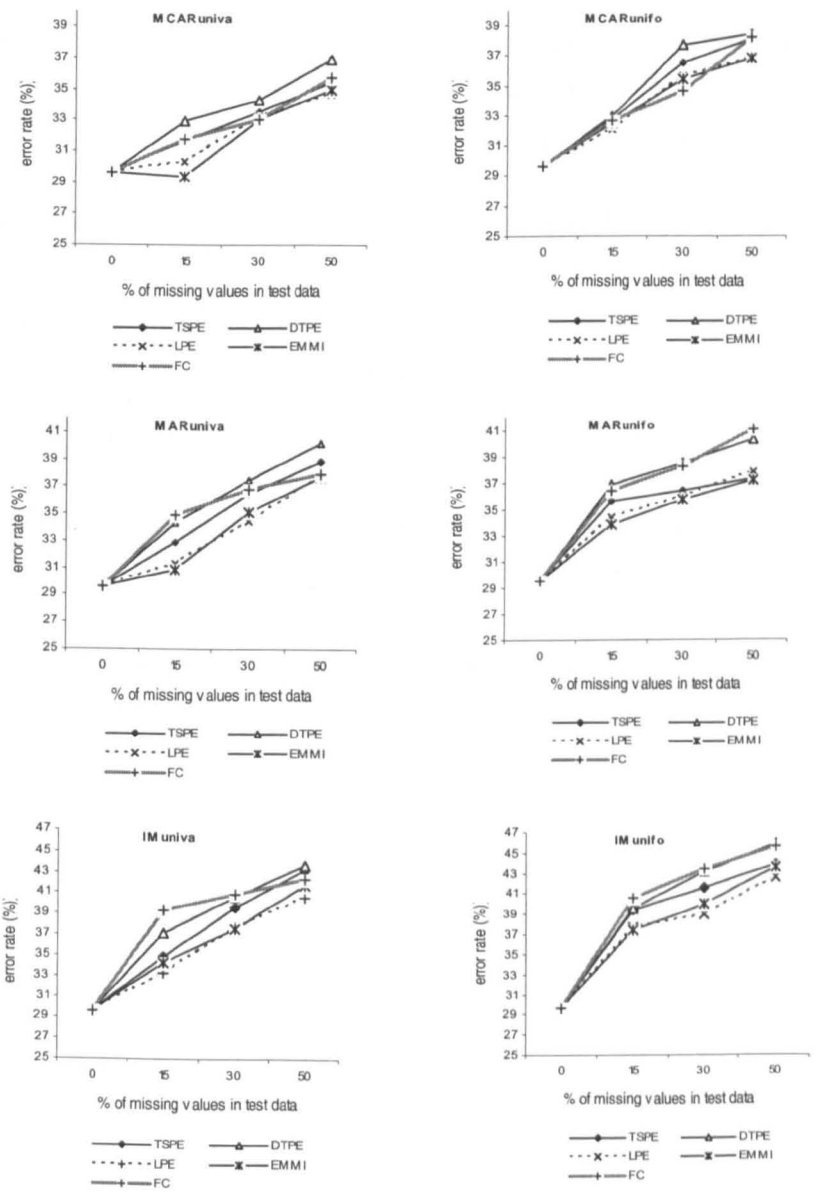

Fig. 6.5. Comparative results of current and new testing methods for the german data. A) MCAR*univa*, B) MCAR*unifo*, C) MAR*univa*, D) MAR*unifo*, E) IM*univa*, F) IM*unifo*

Results for the MCAR*unifo* suite show bigger increases in error rates compared with MCAR*univa* data (Figure 6.5B). In the MAR*univa suite*, EMMI shows a slightly superior overall performance compared with LPE while FC appears to be more robust to MAR*univa* data than MCAR*univa* data (Figure 6.5C). The results for the MAR*unifo* suite (Figure 6.5D) are similar to the ones already observed in the MCAR*unifo* suite. For the IM*univa* suite, the results illustrated in Figure 6.5E show a relatively superior performance by LPE over EMMI. FC becomes more affective with increases in proportion of missingness. In the IM*unifo* case, LPE's overall performance improves with increases in proportion of missingness (Figure 6.5F).

LPE is more robust to missing values for this kind of dataset, especially for IM data. Also, the performance of LPE seems to be better on average when missing values are distributed among all attributes. However, the competitive performance of LPE to methods like EMMI for MAR data is rather surprising. Another surprising result is the poor performance of FC to methods like TSPE and DTPE, especially when missing values are distributed among all attributes. As expected, TSPE outperforms DTPE since this is a reasonably big datasets. with a reasonable number of instances in the training set that are used for estimating the probabilities

6.4.3 Current and New Testing Methods: Processing Time

The two major issues in inductive learning are time spent learning and the classification of novel instances. The training sets are processed until the learning algorithms terminate and then the classification accuracy is measured on the corresponding test sets.

For EMMI, the imputation processing time depends on size of the data matrix, and the number of iterations specified for the iterative algorithm. For the probabilistic methods (TSPE, DTPE, LPE and FC), all branches are explored and the results are combined to reflect the relative probabilities of the different outcomes. Furthermore, for the probabilistic methods, the processing time grows exponentially as a function

of the number of missing attribute values and becomes intractable for large datasets with large amounts of missing data.

The results are based on four datasets; three of the datasets are the largest in the experiments and they encompass purely numerical attributes, purely nominal attributes and mixed attributes, respectively; the fourth dataset is a dataset with mixed attributes whose results were discussed in sub section 6.4.2.2. Only the IM mechanism (a strong condition) and 50% level of missing values combination is considered. Also, only the condition when missing values are distributed in all the attributes is considered. In addition, the testing time is for a whole collection of classifications (depending on the size of the test data) not just one classification.

Table 6.2 shows the approximate running time for current and new testing methods datasets with purely numerical attributes (letter); purely categorical attributes (kr-vs-kp data); and mixed attributes (german and zoo). In addition, the **boldface** font in the table indicates that a method is better than the others.

The quickest method for processing the letter dataset is DTPE, closely followed by TSPE and FC. EMMI has the slowest running time. LPE is about 1.6 times slower than FC but approximately 1.3 times faster than EMMI.

Results for the kr-vs-kp data problem show EMMI, once again, achieving the slowest running time. The quickest running time it now by TSPE which is about 1.3 times quicker than FC and about 3 times quicker than EMMI. EMMI is now 1.6 times slower than LPE while FC is 1.4 times quicker than LPE.

The results in Table 6.2 also show EMMI achieving the slowest running time for the german data problem. Once again, the quickest running time is by TSPE which is 5 times quicker than EMMI. LPE is 1.7 times faster than EMMI and about 1.3 times faster than FC.

Table 6.2 shows TSPE being the quickest method for testing the zoo dataset. FC is two times slower to run compared with DTPE but is 1.2 times faster than LPE. Even though LPE is slower than FC, it is still 1.25 times faster than EMMI.

Table 6.2 Processing time (in seconds) for new and current testing methods for selected datasets

Method	Approximate time on a dataset with purely numerical attributes (s)	Approximate time on a dataset with purely nominal attributes (s)	Approximate time on a dataset with mixed attributes (s)	Approximate time on a dataset with mixed attributes (s)	Description
EMMI	4500	1100	500	90	initial parameter estimates by EM algorithm; iterative simulation; imputation of the missing values
FC	2300	500	400	60	exploration of all branches; summing of weights of instance fragments classified in different ways at leaf nodes
TSPE	3000	**400**	**100**	**18**	exploration of all branches; estimation of probabilities using training data
DTPE	**2000**	600	200	30	exploration of all branches; estimation of probabilities using information on decision tree
LPE	3600	700	300	82	exploration of all branches; estimation of probabilities using binary or multinomial logit models

6.5. Discussion

The main objective on this chapter was to develop a new technique for handling unknown values in the test data and experimentally compare or evaluate the new method with various current methods for handling unknown attribute values using tree-based models. The two current methods previously proposed to deal with the missing value problem are EMMI and FC. The proposed probabilistic methods are TSPE, DTPE and LPE. Hence, the problem stated in Section 6.1 has been solved as shown in Section 6.4.1 and Section 6.4.2, where an algorithm capable of handling missing values in the classification stage in reasonable time has been developed. The principal mechanism needed is one that would improve the estimation of probabilities.

Results of the experiments have already been presented in Figures 6.2 – 6.5. The performance criterion used for the experiments was the classification accuracy required from testing sets.

The comparison with current methods yielded a few interesting results. The experiments showed a model-based approach used for handling incomplete data or unknown attribute values when using decision trees performing better than three probabilistic approaches, especially for datasets with numerical attributes. For datasets with mixed attributes, one of the probabilistic methods was a serious competitor to the model-based approach. In fact, it slightly outperformed the model-based approach in some of the datasets. This is true for all the missing data mechanisms and at different missing value proportions.

Once again, it appears that the main determining factor for missing values techniques, especially for smaller percentages of missing values, is the missing data mechanism. In other words, in datasets with small proportion of instances with missing values there is not much difference between the missing data techniques. However, as the proportion of missing values increases, the distribution of missing values among attributes becomes very important. All the current and proposed methods exhibit bigger error rates when missing values are distributed among all

attributes compared with when the missing values are in only one attribute variable.

When examining to robustness of missing data techniques for tolerating missing values, DTSE is evidently the worst overall method for handling incomplete test data while EMMI has the best overall performance with serious competition from LPE. FC is the third most effective missing data technique. This conclusion is valid for number of attributes with missing values, different amount of missing values, and missing data mechanisms. However, there are some slight differences in performances between some of the current and proposed methods at the 15% level of missing values.

We have shown that estimating the probabilities using the instances in the training data uses much more information than when you are estimating the probabilities using the information taken directly from the decision tree. However, there was a possibility of the latter method performing better since is uses less conditioning on the data. Also, even though you are estimating the 'wrong' probabilities, their estimates are based on more data, and hence are better quality estimates. This is despite the fact that the former is considered to be the correct and 'proper' way of estimating the probabilities but shown not to work very well. The use of less information implies that for the first method to work well you need a lot of training data instances.

An attempt was therefore made to improve the prediction of probabilities, hence, classification accuracy for each method and for each dataset. The binary and multinomial logit models were used to predict these probabilities. The type of model used depended on the class variable of that particular dataset. This was some form of smoothing procedure, hence, probabilistic 'smoothing' method.

The introduction of the binary logit and multinomial logit models to estimate probabilities that are eventually used to predict membership of a class yielded interesting and promising results. First, there was a significant improvement in the results with the LPE method outperforming almost all the other methods with very good accuracy rates (with the exception of the EMMI method). This was the case on

about half of the datasets that had decision trees with large depths (i.e., with many splits and terminal nodes). Secondly, for those datasets with a few splits the method still performed quite comparably to EMMI. This further shows that conditioning of the probabilities, and thus using the training data to predict the probabilities, does have an impact on classification accuracy.

Despite its superior overall performance, EMMI has one very important limitation compared to LPE and the other methods: it is very computer intensive. In the experiments the speed of LPE was found to be about twice as quick to run as EMMI, especially for big datasets and for datasets with a lot of nominal attributes. Thus, not only does the proposed technique deal with missing values almost as well as EMMI it handles incomplete test data at a lower cost.

Also, in contrast to EMMI, the proposed approach does not make representational assumptions or presupposes other models constraints. Therefore, it is suitable for a wide variety of datasets. In the experiments the speed of LPE was found not to be much worse (on average, 1.1 times slower) than FC. Furthermore, from the experimental results of this thesis, the proposed method is recommended to handle incomplete test data for datasets with mixed attributes. The proposed method is also good for dealing with missing values on categorical attributes.

The proposed approach to handling the missing value problem using decision trees is different to other methods. When estimating the probabilities, which are eventually used for predicting a class membership of an instance, all the possible routes that the instance might branch along the decision tree to any respective leaf or terminal node are consulted and considered. Then the probabilities related to each and every route that an instance branches on are calculated at once. Hence, even when another new instance that needs to be classified comes along, one need not repeat the same process of determining the probabilities but just uses the already available information to predict the class of that new instance. This saves a lot of computational time, which is the main strength of this technique.

A quicker algorithm for classifying incomplete vectors by considering two cases has been developed: That either a new instance will branch to the left (first case) or to

the right (second case) of an internal node with non-missing attribute values. For each case three methods that could be used when classifying incomplete vectors using decision trees were proposed. For all methods the main idea is to predict (using basic probability calculations) the class membership for a particular new instance given that there are missing values among some of the attributes. Since the main idea is to predict probabilities that are thus used for predicting the class membership of a particular instance, three approaches to the problem were developed to carry out this task. One approach uses the training data instances to estimate these probabilities; the second uses the information given in the decision tree; and the third uses binary or multinomial logit models to estimate the probabilities.

PAGE

NUMBERING

AS ORIGINAL

Chapter 7

Ensemble Missing Data Methods and Decision Trees

7.1 Introduction

Many works in machine learning and statistics have shown that combining (ensemble) individual classifiers is an effective technique for improving accuracy of classification (Breiman, 1996, Freund *et al.*, 1996, Bauer and Kohavi, 1999). There are different ways in which ensembles can be generated, and the resulting output combined to classify new instances (Dietterich, 2000a). The popular approaches to creating ensembles include changing the instances used for training through techniques such as Bagging (Breiman 1996), Boosting (Freund and Schapire, 1996), and pasting (Breiman 1996), changing the features used in training (Ho 1995), and introducing randomness in the classifier itself (Dietterich 2000b).

This strategy is investigated in the context of decision trees and incomplete data. The basic idea is that an assembly of experts tends to predict better than a single one does. It can be assumed that better performances could be expected as each technique of the ensemble make errors "in different ways". In other words, as an individual missing data technique makes a mistake the other can correct it.

This work proposes a novel ensemble method to improve the robustness and accuracy of tree classifiers when data is incomplete. Combination of class predictions achieved by decision trees using two missing data techniques (EMMI and MIA) that have proven very effective in previous experiments in this thesis are utilised. The new method shall be called EMIMIA (for Ensemble Multiple Imputation and Missing Incorporated in Attributes) and its extended version REMIMIA (for Resampling Ensemble Multiple Imputation and Missing Incorporated in Attributes).

The details of EMIMIA and REMIMIA are presented in Section 7.2. Sections 7.3 and 7.4 present the experimental setup and then the results which show that EMIMIA outperforms both EMMI and MIA, individually, while REMIMIA compares favourably with EMMI and MIA. In addition, not only does REMIMIA appear to have a computational advantage over EMIMIA in terms computational costs, especially for large datasets, it grows smaller trees which are more interpretable.

7.2 Combining Missing Data Techniques Within the Mechanism of Growing and Testing Decision Trees

7.2.1 EMIMIA Technique

Given the experimental results in Chapters 5 and 6 a simple new ensemble method in decision trees and incomplete data is proposed. The new method makes use of all data available and utilises a systematic patterns of classification results based on two methods for handling incomplete training and test data. The new generalized algorithm is summarised in Figure 7.1.

The following example demonstrates the mechanics of the proposed procedure (Table 7.1). Suppose that there are two methods used to handle incomplete data when growing decision trees, resulting in two trees. Also, there are only two classes in the data: 1 and 2. Using the predicted probabilities, there are four possible classification patterns from the training data such as (1, 1), (1, 2), (2, 1) (2, 2). The first element in each pair denotes the class predictions by DT_1 and the second by DT_2. If the instance has predictions (1, 1) for both methods, it is simply assigned class 1 anyway. However, if the instance has predictions (1, 2) it is assigned to the class with the higher overall probability which in this case happens to be 2. And so on.

1. Let T be the incomplete training sample.

2. Construct decision trees on T using EMMI and MIA and call them DT_1 and DT_2, respectively. DT_1 and DT_2 are generated by different algorithms, thus different.

3. Predict a future instance (which could have missing attribute values) based on DT_1 and DT_2.

4. When there is a tie in the predicted probabilities, choose the class with the highest probability or else use a random choice when the probabilities between the two methods are equal.

Fig. 7.1. The EMIMIA algorithm

Table 7.1 An example pattern table

Predicted Probabilities (for each method)	Predicted Class (for each method)	Predicted Class (combining methods)
{(0.6, 0.4); (0.7, 0.3)}	(1, 1)	1
{(0.6, 0.4); (0.3, 0.7)}	(1, 2)	2
{(0.4, 0.6); (0.7, 0.3)}	(2, 1)	1
{(0.4, 0.6); (0.3, 0.7)}	(2, 2)	2

7.2.2 REMIMIA Technique

It is possible to construct many variations of the basic EMIMIA algorithm given that such a technique has the potential of producing large ensembles which may not only be difficult to understand but computationally expensive.

In this section the idea of learning the tree is extended by focusing on ensembles of decision trees that are created with a randomized procedure based on sampling. Randomization can be introduced by using random samples of the training data (as in bagging) and running a tree building algorithm or by randomizing the induction algorithm itself given incomplete data.

The new ensemble method consists of three important components. The first component is the selection of re-sampling technique. The missing data techniques used for building the tree is the second, and the combining method to get predictions is the third. The three components of the new method are presented below.

First, dataset is divided into two training sets of equal size and then grow decision trees using the two different missing data techniques. Testing is carried out using by utilizing systematic patterns of predictions from the tree in each training sample. The new generalised algorithm is summarised in Figure 7.2.

The performance of the proposed ensemble method that utilises resampling of the training data (REMIMIA) will be compared empirically with the other methods, mainly, EMIMIA.

1. Let T be the incomplete training sample and T_1 and T_2 be two random halves of T.

2. Construct decision trees on T_1 and T_2 using EMMI and MIA, respectively, and call them DT_1 and DT_2, respectively. DT_1 and DT_2 are generated by different algorithms on different data, thus different.

3. Predict a future instance (which could have missing attribute values) based on DT_1 and DT_2.

4. When there is a tie in the predicted probabilities, choose the class with the highest probability or else use a random choice when the probabilities between the two methods are equal.

Fig. 7.2. The REMIMIA algorithm

For illustration purposes, the reader is once again referred to Table 7.1.

7.3 Experimental Set-Up

In order to empirically evaluate the performance of EMIMIA and REMIMIA with respect to EMMI and MIA, an experiment is used on the same twenty one datasets used in the experiments in previous chapters. The execution time of the two new ensemble methods against the current and new methods for handling incomplete training and test data is further investigated. In fact, the experiments reported in the previous chapters are repeated with exactly the same experimental settings. A low computational cost when using REMIMIA in terms of the reduced training sample when building the tree using incomplete data is expected.

7.4 Experimental Results

In this section a performance of the empirical study of the two new ensemble methods in comparison to the performance conducted by EMMI and MIA, individually, is reported. The behaviour of these methods is explored for two conditions determining the number of attributes with missing values, four different levels of missing values in training and test sets, and for the MCAR, MAR and IM patterns of missing values.

7.4.1 Overall Results - Ensemble Vs. Current and New Missing Data Techniques

Figure 7.3 lists the comparison results which show the average increase in error rates of the four methods averaged over 21 domains as a function of the percentage of missing attribute values. From the analysis of results displayed in Figure 7.3A, the overall best method for handling MCAR*univa* data is EMIMIA. However, REMIMIA performs almost as well as EMIMIA. Also, the average error of EMIMIA and REMIMIA in the 21 domains increases linearly with the growth of missing data proportions.

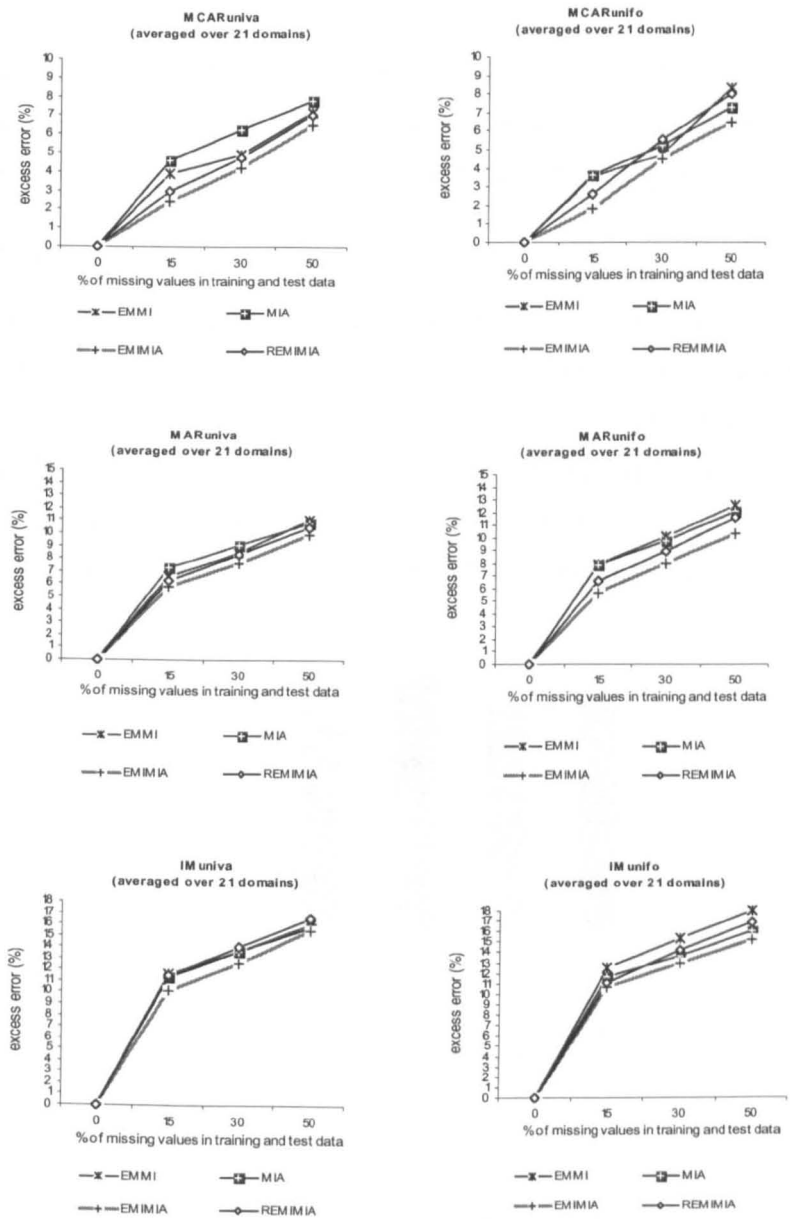

Fig. 7.3. Effects of missing values in training and test data on the excess error for ensemble and missing data methods over the 21 domains. A) MCAR*univa*, B) MCAR*unifo*, C) MAR*univa*, D) MAR*unifo*, E) IM*univa*, F) IM*unifo*

The performance of REMIMIA declines with increasing amount of MCAR*unifo* data (Figure 7.3B). EMIMIA and REMIMIA are the best methods for handling MAR*univa* data (Figure 7.3C). From Figure 7.3D, the performance of EMIMIA and REMIMIA for MAR*unifo* data is very similar to the one observed for MAR*univa* data and

MCAR*univa*. Figure 7.3E present results of methods for IM*univa* data which shows a poor performance by REMIMIA. The results by methods for IM*unifo* data exhibit a similar behaviour to the one observed for IM*univa* data (Figure 7.3F).

Main Effects

All the main effects (methods, number of attributes with missing values, missing data proportions and missing mechanism) were found to be significant at the 1% level of significance as shown in Table 7.2 in the Appendix. Figure 7.4 shows the same trends as Figure 7.3, indicating that the new ensemble method (EMIMIA) has the best level of robustness for tolerating missing values in terms of overall error. EMIMIA is closely followed by EMMI, REMIMIA and MIA, respectively.

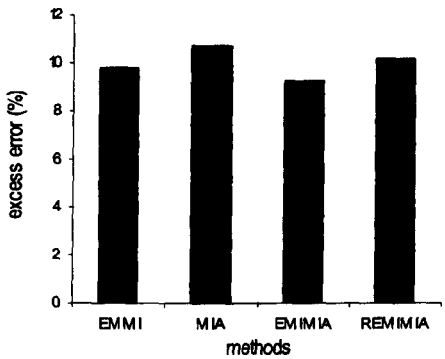

Fig. 7.4. Overall means for current, new and ensemble methods

Interaction Effects

Five two-way interactions were found to be statistically significant at the 1% level (See Table 7.2 in the Appendix) and are, once again, mostly in line with previous results.

7.4.2 Current, New and Ensembles of Missing Data Techniques: Processing Time

The execution time for the two ensemble methods and the current and new methods for handling incomplete training and test data are presented in Table 7.2. Figures in

bold indicate the lowest computational cost by each method and for each dataset. In addition, figures in parentheses indicate execution time for testing.

As expected, EMIMIA suffers from high computational cost compared with the other methods. When growing and testing using the tree, EMMI is about 1.7 and 2.5 times slower to run compared to EMMI and MIA, respectively. This is the case for all the datasets.

When using the tree (testing), the execution times by methods vary by dataset. In fact, the execution time required by EMIMIA is low for the dataset with purely numerical attributes and high for the dataset with mixed attributes.

First, compared with EMIMIA, EMMI and MIA are about 2.9 and 4.8 times faster to run, respectively. This is for the dataset with mixed attributes. Second, for the dataset with purely numerical attributes EMIMIA is about 1.3 and 1.7 times slower compared with EMI and MIA, respectively. Finally, EMIMIA has the slowest running time for the datasets with purely nominal attributes.

Consistent with the re-sampling of the training set technique discussed in Section 7.2, the execution time required by REMIMIA is indeed lower than EMIMIA for all the datasets. In fact, EMIMIA requires about 1.5 times more execution time than REMIMIA for the dataset with purely nominal attributes. Also, REMIMIA is not much worse than EMMI in terms of computational cost.

For the datasets with purely numerical and mixed attributes, REMIMIA is about 1.2 and 1.4 times quicker than EMIMIA, respectively. Also, the execution time for testing when using trees is shorter for REMIMIA than EMIMIA for all the datasets.

EMIMIA is about 1.6 times slower to run for the datasets with purely nominal attributes and about 1.2 and 1.3 times slower to run for the datasets with mixed and purely numerical attributes. For both datasets, REMIMIA is slower than EMMI alone.

For the dataset with purely numerical attributes, both REMIMIA and EMMI achieve the same execution time when testing.

Table 7.2 Processing time (in seconds) of growing and testing trees for current, new and ensemble methods for selected datasets

Method	Approximate time on a dataset with purely numerical attributes (s)	Approximate time on a dataset with purely nominal attributes (s)	Approximate time on a dataset with mixed attributes (s)	Description
EMMI	31500 (4500)	7100 (1100)	4900 (500)	initial parameter estimates by EM algorithm; iterative simulation; imputation of the missing values
MIA	**21600 (3600)**	**5250 (750)**	**3300 (300)**	search through all attributes 1. search through split points and send 'missing' to the left and choose the best split; 2. search through split points and send 'missing' to the right and choose the best split; 3. search through split points and send 'missing' to the left and the rest to the right choose the best split of 1 to 3 and best attribute
EMIMIA	53100 (6000)	12350 (2400)	8200 (1440)	Combination of EMMI and MIA descriptions (without resampling of training data)
REMIMIA	46000 (4500)	8300 (1500)	6000 (1200)	Combination of EMMI and MIA descriptions (with resampling of training data)

7.5. Discussion

In this chapter, two novel ensemble approaches for handling incomplete data in the learning and application phases of tree-based modelling are proposed. These new methods utilize systematic patterns of predictions from decision trees built and used from incomplete data. However, while one of the methods grows the decision trees from the whole training data, the other re-samples the training data before growing the trees. The developed algorithms were compared with two missing data methods using real-world datasets.

Experimental results shown in Figures 7.4 and 7.5 indicate that EMIMIA handles missing attribute values better than EMI and MIA and with improved classification performance. The results also indicate that REMIMIA is as good as EMMI, at least when missing values are only in one attribute.

Comparison between EMIMIA and REMIMIA shows a more consistent performance for the former. The improved performance is probably due to the fact that EMIMIA uses all the training data for each missing data method when growing the trees and the combining of probabilities reduces the variance that would arise in a single method. Another reason follows from the fact that larger samples normally yield consistent results or much more stable model specifications (Godfrey, 1988). In addition, splitting of the training data into two sets by REMIMIA reduces the sample size on which each tree is grown, hence, each tree losing its classification accuracy.

Despite its strengths, EMIMIA consistently takes more time to train and test, especially for large datasets. Furthermore, EMIMIA produces more complex ensembles (in the form of larger trees). However, by using REMIMIA the computational cost was kept low and the trees grown were much smaller and more interpretable. These are two important advantages of REMIMIA over EMIMIA. In addition, for the datasets with purely nominal and mixed attributes, REMIMIA is not much worse than EMMI in terms of computational cost.

Another important trend shows that increasing the number of attributes with missing values seems to have more impact on the ensemble methods than on individual missing data methods. Also, given that the classification performance

of each method varies by mechanism of missing data, it appears that the treatment of missing values not only heavily depends on the missing data proportions but on the nature of the missing data pattern.

We have proposed two simple, novel and yet effective ensemble methods for building and testing decision trees using incomplete data. The proposed methods combine the advantages of EMI and MIA for classification improvement when using decision trees. These methods which appear to give good results are easier to understand even for people without previous exposure to advanced machine learning and statistics topics.

Chapter 8

Concluding Remarks

In this last chapter the major problem considered at the beginning of the thesis – the problem of missing data and decision trees is reviewed. Some of the work and results of the analysis presented in this thesis are summarised. In addition, major contributions of new knowledge of this work in the area of machine learning and statistics are summarised. Furthermore, the main conclusions in the study are presented.

Several current and efficient methods to learn a Decision Tree (DT) from data were looked at. However, these methods were often limited to the assumption that data are complete. The objective underlying this thesis was to develop and evaluate alternative techniques for handling incomplete data in the learning and application phases of tree based modelling. This was carried out on twenty one data sets, for two different patterns of missing data, for three different missing data mechanisms and at various proportions of missing values. Information about the similarities and differences among the methods in their conclusions should be useful to researchers struggling to handle missing values in the case of DTs.

The experimental results in Chapters 4 through 7 allow us to draw conclusions and recommendations about growing and testing decision trees using incomplete data. These are given in the next sections.

8.1 Research Findings

8.1.1. Current Methods

Based on the findings in Chapter 4.2 most of the statistical methods performed reasonably well on all data sets. In fact, there are no clear differences in predictive accuracy between methods for small proportions of missing values. However, as the proportion of missing values increases, the differences in accuracy between methods

begin to appear. In addition, higher levels of missing values are associated with higher classification error rates, and vice versa. Hence, lower levels of missing data (usually 5% or less) are seldom problematic and might not require a great deal of attention by researchers.

As it turns out, EMMI performed better than expected as a method for handling both training and test data in almost every situation. However, this is not to say that EMMI is a silver bullet, but rather a versatile approach. The superior performance of EMMI could have been attributed to the profit it gets from averaging the resulting trees which causes a reduction in variance.

If the sample size is small and massively missing, EMSI and EMMI are highly recommended. EMMI is also highly recommended for the creation of imputed samples and aggregate results from each imputed data set.

Clearly, LD is one of the worst methods for handling incomplete data using decision trees but could still be used for MCAR data especially if the sample size is large. The worst performance of the LD method is quite expected due to the loss of information in discarding incomplete cases, i.e., dropping all the instances with missing observations from the computations. It is possible that the omitted instances carry important information on the relation between the attributes, which the missing value mechanism allows us to use.

CCSI can be recommended for datasets with few classes but only for building trees from incomplete data. Having many classes reduces the sample on which you are imputing and this could be the reason why CCSI struggles with datasets that have many classes for the response variable. MMSI is a good method when missingness occurs in the majority class with the attribute taking on some major value and may be a good method for handling mixed attributes. In fact, for data sets with mainly nominal attributes and with many missing values, MMSI is a reasonable choice to use.

DTSI is recommended when dealing with nominal attributes and when missing values are not in all attributes. DTSI could also be suitable in domains where the lowering in classification error is worth the increase in computational cost.

SVS seems to be effective as a method for handling incomplete test data and for data sets where the correlations between attributes are high. If the missing attributes often occur in the majority class, and the seen features imply a different class that the instances fall into, FC is recommended.

Machine learning methods (FC, DTSI and SVS) are expected to achieve higher accuracy than statistical methods because of their complicated processing. However, only FC showed a somewhat better predictive performance in the experiments even though it is a computationally more demanding approach, i.e., it took more time in processing than most of the statistical methods (with the exception of EMMI).

Some authors (Little and Rubin, 1987; Graham and Donaldson, 1993; Roth, 1994; Tabachnick and Fidell, 2001) have argued that the performance of any missing data technique depends heavily on the mechanisms that lead to missing values. In this work the major role the pattern and mechanism of missing data (how and why the data are missing) plays in the performance of a tree algorithm has been observed. There was sufficient evidence to show that the IM mechanism has more severe impact on predictive performance than the other two mechanisms when both the training and test data have missing values. As the experiments suggest, the MCAR mechanism appears to have less impact on performance than both the MAR and IM mechanisms. The difference in performance between the MCAR and MAR conditions was small, which may explain why the MAR condition is often considered a 'special case' of MCAR. Also, missing values appear to have more impact on classification accuracy when they are distributed among all attributes than when they are only in one attribute.

Also, it was observed that the impact of missing values depends not just upon whether or not a method is for handling incomplete training or test data (individually) but upon a combination of the two. Therefore, neither can be considered in isolation.

The missing data techniques tend to yield high accuracy rates for data sets with few classes compared with data sets with many classes.

8.1.2 Current Vs. New Methods

As stated in Chapter 5, an algorithm, MIA, which is capable of handling both incomplete training and test data in a reasonable time using decision trees, was proposed. The method involves using a set of binary splits whereby each respective split treats the 'missingness' of the data as an important aspect for feature selection. The principal mechanism needed in the improved binary splitting idea was being able to determine the cut-off points among the missingness of the data for each of the three binary splits. Each of the three splits accommodates 'missingness' in the branching; two of them accommodate the missingness of the data along with actual values and one missingness and non-missingness of the data as a dichotomy. Its performance is experimentally tested on twenty one datasets and compared with other different approaches previously proposed to deal with the problem.

When examining the robustness of missing data techniques for tolerating missing values, the proposed procedure shows promise by giving EMMI serious competition in most of the datasets and some of the time outperforming EMMI especially on datasets with purely nominal attributes and mixed attributes. Also, the new algorithm greatly improves its accuracy on IM data when both the training and test data contains a large proportion of missing values.

It was shown experimentally that in general, the new method is much superior to FC. Furthermore, the computational speed of the new method was found not to be much worse (on average 1.2 times slower) than FC but had a computational advantage over EMMI, especially for big datasets and when the fraction of missing data was large. Since data sets in the real world are getting bigger by the day, reducing processing cost is one of the most important problems in the machine learning and statistics field. Thus, not only does the new technique deal with missing values as well and complicated as EMMI, it handles incomplete training and test data at a lower cost.

8.1.3 A Further Idea for Classifying Incomplete Vectors Using Trees

Missing values were found to have more impact when they occur in the test set. Hence, developing a technique that could be used for classifying incomplete test vectors using decision trees based on complete data was another aim of this thesis. Chapter 6 focussed on developing such a technique.

The principal mechanism behind the algorithm is probability theory, and has three approaches to the problem. These approaches are prediction of probabilities using training data (TSPE), estimation of probabilities using decision trees (DTPE), and prediction of probabilities using binomial and multinomial logit models (LPE).

The use of less information implies that for TSPE to work well a lot of training data instances are needed whereas for DTPE the information given is always enough. Hence, the strength of the latter method lies in its ability to use as much information as possible when estimating the probabilities that are used when classifying a new instance. But, the new method, particularly DTPE, initially produced unsatisfactory results. However, the results improved considerably after modification of the technique. Binary and multinomial logit models were used as procedures for predicting class membership of an unknown instance. This was particularly the case on the datasets with mixed attributes and purely nominal attributes and datasets that had decision trees with large depths (i.e., with many number of splits and terminal nodes). Also, for those data sets with a few splits the method still performed quite comparably to EMMI. This further shows that conditioning of the probabilities, and thus using the training data to predict the probabilities, does have an impact on classification accuracy.

Although the new approach to handling the missing value problem using decision trees is superficially similar to Quinlan (1993)'s FC approach, there are some slight but very crucial differences which are given in Chapter 5. For an example, during the process of classifying a new instance, FC exploits the first valid path/rule while the proposed procedure considers all the valid paths/rules and then deduces a more

consensual result. This saves a lot of computational time, which is one of the main strengths of this technique. Also, the new method is much superior to FC, especially for test sets with larger percentages of missing attribute values.

Once again, despite its superior overall performance, EMMI has one very important limitation compared to LPE: it is very computer intensive. In fact, LPE is about twice as fast to run as EMMI, especially for big data sets and with a lot of missing values.

8.1.4 Ensemble Methods

On the classification accuracy improvement of decision trees when using current and new methods for handling incomplete data, two new ensemble methods that are based on resampling and non- resampling of the training data, respectively, were proposed in Chapter 7. The algorithms take boosting (without resampling) and bagging (with resampling) style ensemble approaches by combining two missing data techniques (EMMI and MIA) to engender performance improvement over one technique.

Experimental results on 21 real-world datasets show the ensemble method without re-sampling of the training data (EMIMIA) as performing best and yielding improved accuracy mean classification rates compared with effective individual methods for handling missing data and the ensemble method with resampling.

According to the experiments, the computational disadvantage of the ensemble method without re-sampling was identified, especially when dealing with larger datasets. This led to the development of the second ensemble method (REMIMIA) that is based on resampling the training data. It was found that splitting the training data into two sets and then growing a decision tree from each set using the two missing data techniques played a key role in time efficiency while maintaining comparable accuracy with EMMI and MIA.

8.2 Summary of Contributions

The contributions of this research pertain specifically to incomplete data and decision trees. Some original demonstrations and ideas in the thesis are given below:

1. The first contribution made by this thesis was the addressing of a very important topic in machine learning and statistics: missing data and tree-based models. Both research communities are beginning to recognize the criticality of developing techniques for handling incomplete data when using decision trees.

2. The second contribution was the development of a new technique for handling both incomplete training and test data. This technique exploits the properties of split selection measures by using 'missing' as a pseudo value. While this technique did not always outperform other techniques, it has been shown that considering 'missingness' as an important factor can yield very good results, especially for IM data. Given that IM data is difficult to deal with, this provides a base for further research in the area.

3. A much quicker method with three approaches for handling incomplete test data has been developed. The use of basic probability calculations for handling incomplete test data has been demonstrated experimentally.

4. A fourth contribution of the thesis is two simple, novel, yet effective ensemble techniques for building and testing decision trees using incomplete data. The proposed methods combine the advantages of EMI and MIA, taking bagging and boosting-style ensemble approaches. While the ensemble method without resampling of the training data led to significant performance gains in most of the experiments it was computationally expensive compared with the ensemble method with resampling.

5. This research has also applied current techniques for handling unknown attribute values when using trees. This is the biggest comparative study of the performance of the different approaches previously proposed to deal with the missing (unknown) attribute values problem for tree-based models on

classification accuracy to be carried out. Prior to this, little to no research suggesting the use of these techniques has been published.

6. Lastly, this research compared these different techniques and made several applications based recommendations for the implementation of these methods. One of these recommendations is to exert efforts to collect data to the fullest extent and of highest quality. By so doing researchers keep missing data to a minimum and therefore reduce bias and distortion in estimating population parameters or testing pertinent hypothesis. Also, researchers must determine whether the cause and pattern of missing data will seriously impair the quality of inferences derived and which procedure, if any, is most appropriate for handling missing data. A careful examination of factors causing missing data and missing data pattern allows researchers to decide if and how best to deal with missing data in a study.

In sum, this thesis provides the beginnings of a better understanding of the relative strengths and weaknesses of techniques for extracting decision trees from possibly incomplete databases. It is hoped that it will motivate future theoretical and empirical investigations into missing data and decision trees.

References

Arbuckle, J.L. (1996a) Full information estimation in the presence of incomplete data. In: G.A. Marcoulides & R.E. Schumacker (Eds.) Advances structural equation modelling. Mahwah, NJ: Lawrence Erlbaum Publishers

Arbuckle, J. (1996b). *Amos Users Guide: Version 3.6.* Small Waters Corp., Chicago.

Agrawal, R., Imielinski, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. In *Proc. of the ACM SIGMOD Conference on Management Data*, Washington, D.C.

Agresti, A. (1990). *Categorical Data Analysis.* New York: John Wiley.

Aha, D.W., Kibler, D., and Albert, M.K. (1991). Instance-based learning algorithms, *Machine Learning,* **24**, 173-202.

Afifi, A. and Elashoff, R.M. (1966). Missing observations in multivariate statistics I. Review of the literature, *Journal of the American Statistical Association*, **61**, 595-604.

Allison, P.D. (2001). Missing Data. Thousand Oaks, CA: Sage.

Backer, G. (1996). Learning with missing data using Genetic Programming. *The 1st online Workshop on Software Computing* (WSC1), 19-30, Nogaya, Japan.

Bandyopadhyay, S., Murthy, C.A., and Pal, S.K. (1995). Pattern classification with genetic algorithms, *Pattern Recognition Letters*, **16**, 801-808.

Batista, G. and Monard, M.C. (2003). An Analysis of Four Missing Data Treatment Methods for Supervised Learning, *Applied Artificial Intelligence,* **17**, 519-533.

Bauer, E., and Kohavi, R. (1999). An empirical comparison of voting classification algorithms: Bagging, boosting and variants. *Machine Learning*, **36**, 1/2, 105-139.

Beale, E.M.L. and Little, R.J.A. (1975). Missing values in multivariate analysis. *Journal of the Royal Statistical Society*, **37** (Series B), 129-145.

Becker, R., Chambers, J., and Wilks, A. (1988). *The New S Language*. Wadsworth International Group.

Bernado, J. and Smith, A. (1994). *Bayesian Theory*. Chichester: John Wiley.

Bishop, Y.M.M., Fienberg, S.E., and, Holland, P.W. (1975). *Discrete multivariate analysis: theory and practice*. MIT Press, Cambridge, Massachusetts.

Biggs, D., de Ville, B., and Suen, E. (1991). A method for choosing multiway partitions for classification and decision trees. *Journal of Applied Statistics*, **18**, 49-62.

Blake, C., Keogh, E., and Merz, C.J. (1999). UCI respiratory of machine learning databases. University of California, Irvine, Department of Information and Computer Sciences. http://www.ics.uci.edu/~mlearn/MLRepository.html.

Bohanec, M., and Bratko, I. (1994). Trading accuracy for simplicity in decision trees. *Machine Learning*, **15**, 223-250.

Bratko, I. and Kononenko, I. (1986). Learning diagnostic rules from incomplete and noisy data. *AI Methods in Statistics*, UNICOM Seminar, London, December 1986.

Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). *Classification and Regression Trees*, Wadsworth.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, **26** (2), 123-140.

Brodley, C.E., and Utgoff, P.E. (1995). Multivariate Decision Trees. *Machine Learning*, **19**, 45-77.

Brown, R.L. (1994). Efficacy of the indirect approach for estimating structural equation model with missing data: A comparison of methods. *Structural Equation Modelling: A Multidisciplinary Journal*, **1**, 287-316.

Bruha, I. and Franek, F. (1996). Comparison of various routines for unknown attribute value processing: the covering paradigm. *International Journal of Pattern Recognition and Artificial Intelligence*. **10** (8), 939-955.

Buck, S.F. (1960). A method of estimation of missing values in multivariate data suitable for use with an electronic computer. *Journal of the Royal Statistical Society, B*, **22**, 302-307.

Buntine, W. (1990). *A Theory of Learning Classification Rules.* PhD dissertation, University of Sydney, Australia.

Buntine, W. (1991). Classifiers: A theoretical and empirical study: In *International Joint Conference on Artificial Intelligence,* Sydney, Morgan Kaufmann, 638-644.

Buntine, W.L. (1992). Learning Classification Trees, *Statistics and Computing,* **2,** 63-73.

Buntine, W. (1994). Operations for Learning with Graphical Models. In *Journal of Artificial Intelligence Research,* **2,** 638-644.

Buntine, W. and Niblett, T (1992). A further comparison of Splitting Rules for Decision Tree Induction. *Machine Leaning,* **8,** 75-85.

Burges, C.J.C. (1998). A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery.* Kluwer Academic Publishers.

Byrne, B. N. (2001). *Structural Equation Modelling with AMOS.* Rahwah, NJ: Lawrence Erlbaum Associates.

Cartwright, M., Shepperd, M.J., and Song, Q. (2003). Dealing with Missing Software Project Data. In *Proceedings of the 9th International Symposium on Software Metrics 2003,* 154-165.

Cestnik, B., Kononenko, I. and Bratko, I (1987). Assistant 86: a knowledge-elicitation tool for sophisticated users. In I. Bratko and N. Lavrac, editors, *European Working Session on Learning* – EWSL87. Sigma Press, Wimslow, England, 1987.

Cestnik, B., and Bratko, I (1991). On estimating probabilities in tree pruning. In *Proceedings of the 5th European Workshop Session on Learning.* 138-150. Porto, Portugal: Springer-Verlag.

Chipman, H., George, E.I., and MuCulloch, R.E. (1998a). Bayesian CART Model Search (with discussion). *Journal of the American Statistical Association,* **93,** 935-960.

Chipman, H., George, E.I., and MuCulloch, R.E. (1998b). Hierarchical Priors for Bayesian CART Shrinkage, Working Paper 98-03, Dept. of Statistics and Actuarial Science, University of Waterloo.

Chiu, D.K.Y. and Wong, A.K.C. (1986). Synthesizing knowledge: A cluster analysis approach using event covering, *IEEE Transactions on Systems, Man and Cybernetics*, **16** (2), 251-259.

Chiu, H.Y. and Sedransk, J (1986). A Bayesian procedure for imputing missing values in sample surveys. *Journal of the American Statistical Association*, **81** (395), 667-675.

Chou, P. (1991). Optimal Partitioning for Classification and Regression Trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **13** (4) 340-354.

Clark, P. and Niblett, T. (1989). The CN2 Induction algorithm. *Machine Learning*, **3**, 261-283.

Clark, P. and Boswell, R. (1991). Rule Induction with CN2: Some recent improvements. In *Proceedings of the Sixth European Workshop Session on Learning*, (151-163). Porto, Portugal: Springer Verlag

Clark, L., and Pregibon, D. (1991). Tree-Based Models. In *Statistical models in S*, J. Chambers and T Hastie, Eds., Wadsworth, 377-420.

Cohen, W.W. (1995). Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning*. Lake Tahoe, California.

Cohen, J. and Cohen, P. (1983). *Applied multiple regression/correlation analysis for the bahavioral sciences* (2nd Ed.). Hillsdale, NJ: Lawrence Erlbaum Associates.

Cohen, P.R. and Jensen, D. (1997). Overfitting explained. In *Proceedings of the 6th International Conference on Machine Learning,* (115-123). Tahoe City, CA: Morgan Kauffman.

Cohen, J. and Martin. F. (1997). Numerical Taxonomy on data: experimental results. In *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms,* (410-417). New Orleans, LA.

Collett, D. (1991). *Modelling Binary Data*. Chapman and Hall.

Conversano, C. and Siciliano, R. (2002). Tree Based Classifiers for Conditional Incremental Missing Data Imputation. A *Conference for dealing with erroneous and missing data.* Jyväskylä, *Fi*nland.

Cooper, G.F. (1990). The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence,* **42**, 393-405.

Cooper, G., and Herskovitz, E. (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, **9**, 309-347.

Cover, T.M., and Hart, P.E. (1967). Nearest neighbour pattern classification. *IEEE Transactions on Information Theory,* **13**, 1, 21-27.

Cox, D.R. (1966). Some procedures associated with the logistic qualitative response curve. In *Research papers in Statistics: Festschrift for J. Neyman* (ed. F.N. David), Wiley, New York, 55-71

Cox, D R., and Wermuth, N. (1966). *Multivariate Dependencies.* Chapman and Hall, London.

Cox, L.A., Qui, Y., and Kuehner, W (1989). Heuristic least-cost computation of discrete classification functions with uncertain argument values. *Annals of Operations Research,* **21** (1), 1-30.

Crawford, S.L. (1989). Extensions to the CART algorithm. *International Journal of Man-Machine Studies,* **31** (2), 197-217.

Dasrathy, B.V. (1980). Nosing around Neighbourhood: A New System Structure and Classification Rule for Recognition in Partially Exposed Environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **2**, 1, 67-71.

Dasrathy, B.V. (1991). Nearest Neighbour (NN) Norms: NN Pattern Classification Techniques. *IEEE Computer Society Press*, Los Alamitos, California.

Day, N.E. and Kerridge, D.F. (1967). A general maximum likelihood discriminant. *Biometrics,* **23**, 313-323.

Dempster, A.P., Laird, N.M., and Rubin, D.B. (1977). Maximum likelihood estimation from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, **39**, 1-38.

Denison, D.G.T., Mallick, B.K., and Smith, A.F.M. (1998). A Bayesian CART Algorithm. *Biometrika*, **85** (2), 363-377.

Dillon, W. and Goldstein, M. (1984). *Multivariate Analysis, Methods and Applications.* New York: John Wiley.

Dietterich, T. (2000a). Ensemble methods in machine learning. In *Proceedings of the First International Workshop on Multiple Classifier Systems*, Springer Verlag, 1-15.

Dietterich, T. (2000b). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, **40** (2), 139-158.

Dixon, W. J., Brown, M.B., Engelman, L., Frane, J.W., Hill, M.A., Jennrich, R.I., and Toporek, J.D. (1983). *BMDP Statistical software.* Berkeley: University of California Press.

Domingos, P and Pazzani, M. (1996). Beyond independence: conditions for the optimality of the simple Bayesian classifier. In *Proceedings of the 13th International Conference on Machine Learning,* (105-112), Bari, Italy.

Duda, R., and Hart, P. (1973). *Pattern Classification and Scene Analysis.* New York: John Wiley.

Edwards, D. (1995). *Introduction to Graphical Modelling.* Springer-Verlag, New York.

Efron, B. (1982). *The jacknife, the bootstrap and other resampling plans.* Philadelphia: Society for Industrial and Applied Mathematics.

Enders C.K (2001). The Performance of the Full Information Maximum Likelihood Estimator in Multiple Regression Models With Missing Data. *Educational and Psychological Measurement*, **61** (5), 713-740.

Ernst, L.R. (1980). Variance of the estimated mean for several imputation procedures. *Proceedings of the Survey Research Section, American Statistical Association*, 716-720.

Esposito, F., Malerba, D., and Semeraro (1993). Decision tree pruning as a search in the state space. In P. Brazil, editor, *Machine Learning: ECML-93*. LNAI 667, Springer Verlag.

Esposito, F., Malerba, D., and Semeraro (1995). A further study of pruning methods in decision tree induction. In *Proceedings of the 5th International Workshop on Artificial Intelligence and Statistics,* (211-218). Ft. Lauderdale, FL.

Esposito, F., Malerba, D., and Semeraro (1997). A comparative analysis of methods for pruning decision trees. *Transactions on Pattern Analysis and Machine Intelligence.* **19** (5), 476-491.

Everitt, B.S. (1984). *An Introduction to Latent variable models*. Chapman and Hall.

Everitt, B.S. and Hand, D.J. (1981). *Finite Mixture Distributions*. Chapman and Hall, London.

Everitt, B.S. and Dunn, G. (1991). *Applied Multivariate Data Analysis.* Edward Arnold Publishers.

Farhangfar, A., Kurgan, L. and Pedrycz, W. (2004). Experimental Analysis of Methods for Handling Missing Values in Databases, *Intelligent Computing: Theory and Applications II Conference*, held in conjunction with the *SPIE Defense and Security Symposium* (formerly AeroSense), Orlando, FL.

Fayyad, U.M. (1991). *On the Induction of Decision Trees for Multiple Concept Learning*. PhD dissertation, EECS Dept., The University of Michigan.

Fayyad, U.M. and Irani, K.B. (1991). A Machine Learning Algorithm (GID3*) for Automated Knowledge Acquisition: Improvements and Extensions. *General Motors Report CS-634.* Warren, MI: GM Research Labs.

Fayyad, U.M. and Irani, K. B. (1993). Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, 1022-1027. Morgan Kaufmann.

Feelders, A.J. (1999). Handling Missing Data in Trees: Surrogate Splits or Statistical Imputation? *In Proceedings of the 3rd European conference on principles and practice of knowledge discovery in data bases* (PKDD99), (329-334). Springer Verlag.

Fisher, R.A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics,* **7**, 179-188.

Ford, B.M. (1983). *An Overview of Hot Deck Procedures. Incomplete Data in Sample Surveys.* **2**. New York: Academic Press.

Forsyth, R. and Rada, R. (1986). *Machine Learning: Applications in expert systems and information retrieval.* Sellis Horwood Limited, Chichester.

Forsyth, R.S., Clarke, D.D., and Wright, R.L. (1994). Overfitting revisited: an information-theoretic approach to simplifying discrimination trees. *Journal of Experimental and Theoretical Artificial Intelligence,* **6** (3), 289-302.

Frank, E. and Witten, I.H. (1999). Making Better Use of Global Discretization, Proceedings of the Sixteenth International Conference on Machine Learning, 115-123.

Friedman, F.H. (1977). A recursive partitioning decision rule for non-parametric classification. *IEEE Transactions on Computers,* 404-408.

Freund, Y. and Schapire, R. (1996). Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the Thirteenth International Conference,* 148-156.

Friedman, F.H., Kohavi, R and Yun, Y. (1996). Lazy decision trees. In *Proceedings of the 13th National Conference on Artificial Intelligence,* (717-724), AAI, Press/MIT Press.

Fujikawa, Y., Ho, T.B. (2002). Cluster-based Algorithms for Filling Missing Values, 6th Pacific-Asia Conf. Knowledge Discovery and Data Mining, Taiwan, 6-9 May, Lecture Notes in Artificial Intelligence 2336, Springer Verlag, 549-554.

Gammerman, A., Lou, Z., Aitken, C.G.G. and Brewer, M.J. (1995). Exact and approximate algorithms and their implementation in mixed graphical models. In Gammerman, 33-55.

Gediga, G. and Düntsch, I. (2003). Maximum Consistency of Incomplete Data via Non-Invasive Imputation. *Artificial Intelligence Review*, **19** (1), 93-107.

Gelfand, S.B., Ravishankar, C.S. and Delp, E.J. (1991). An iterative growing and pruning algorithm for classification tree design. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* **13**, 163-74.

Gelman, A. and Rubin, D.B. (1992). Inference from iterative simulation using multiple sequences. *Statistical Science*, **7** (4), 457-472.

Geman, S. and Geman, D. (1984). Stochastic Relaxation, Gibbs Distributions and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **6**, 721-741.

Gersho, A. and Gray, R.M. (1991). *Vector Quantization and Signal Compression.* Kluwer Academic Pub.

Gilks, W.R., Richardson, S., and Spiegelhalter D. J. (1996). *Markov Chain Monte Carlo in practice.* Chapman and Hall, London.

Gillo, M.W. (1972). MAID: A Honeywell 600 program for an automatised survey analysis. *Behavioral Science*, **17**, 251—252.

Gleser, M.A. and Collen, M.F. (1972). Towards automated medical decisions. *Comp. and Biomedical Research.* **5** (2), 180-189.

Godfrey, L., *Misspecification Tests in Econometrics*, Cambridge University Press, 1988.

Gordon, A.D. (1981). *Classification.* New York: John Wiley and Sons.

Gorman, R. P. and Sejnowski, T. J. (1988). Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets in Neural Networks, **1**, 75-89.

Gower, J.C. (1998). Classification Overview, In *Encyclopaedia of Biostatistics*, **2**, 656-667, Wiley.

Graham, J.W. and Donaldson, S.I. (1993). Evaluating interventions with differential attrition: the importance of nonresponse mechanisms and use at follow up data. *Journal of Applied Psychology*, **78**, 119-118.

Graham, J. W., Hofer, S. M., Donaldson, S. I., MacKinnon, D. P., and Schafer, J. L. (1997). Analysis with missing data in prevention research. In K. J. Bryant, M. W. Windle, and S. J. West (Eds.). *The science of prevention: Methodological advances from alcohol and substance abuse research* (325-366). Washington, DC: American Psychological Association.

Grefenstette, J. J. (1991). Strategy acquisition with genetic algorithms. In Handbook of Genetic Algorithms, L. D. Davis (Ed.), Boston: Van Nostrand Reinhold.

Grzymala-Busse, J.W. (1991). On the unknown attribute values in learning from examples. In *Proceedings of the ISMIS-91, 6th International Symposium on methodologies for Intelligent Systems*, Charlotte, North Carolina, October 16-19, (368-377), *Lecture notes in Artificial Intelligence*, Vol. 542, Springer Verlag, Berlin, Heidelberg, New York.

Grzymala-Busse, J.W. and Hu, M. (2000). A Comparison of Several Approaches to Missing Attribute Values in Data Mining. In *RSCTC '2000*, 340-347, Banff, California.

Hand, D.J. (1981). *Discrimination and Classification*. New York: John Wiley and Sons.

Hand, D.J. (1982). *Kernel Discriminant Analysis.* Chichester, UK: Research Studies Press (John Wiley and Sons).

Hand, D.J. (1997). *Construction and Assessment of Classification Rules*. John Wiley and Sons, Chichester.

Hand, D.J. (2000). Private communication.

Hand, D.J., Mannila, H., and Smyth, P. (2001). *Principles of Data Mining*. MIT Press.

Hand, D.J., Blunt, G., Kelly, M.G., and Adams, N.M. (2000). Data mining for fun and profit. *Statistical Science*, **15**, 111-131.

Hart, A.E. (1984). Experience in the Use of an Inductive Learning System in Knowledge Engineering: In Brammer, Max (ed.) *Research & Development in Expert Systems:* Cambridge: Cambridge University Press.

Hartley, H.O. and Hocking, R.R. (1971). The analysis of incomplete data. *Biometrics*, **27**, 783-823.

Haussler, D. (1988). Quantifying inductive bias: AI learning algorithms and Valiant learning. *Artificial Intelligence*, **36** (2): 177-222.

Hyafil, L. and Rivest, R.L. (1976). Constructing optimal binary trees in NP-complete. *Information Processing Letters*, **5** (1), 15-17.

Hayishi, T., Bastian, A., and Jain, L.C. (1998). Generation of fuzzy decision trees by fuzzy ID3 with adjustments of AND/OR operators. In *Proceedings of the 1998 International Conference on Fuzzy Systems (FUZZ-IEEE 98),* (681-685). NJ: IEEE Press.

Heckerman, D. (1996). A tutorial on learning Bayesian networks. *Technical Report* MSR-TR-95-06, Microsoft, Redmond, WA. Revised January 1996.

Heckerman, D., Geiger, D., and Chickering, D.M. (1994). Learning Bayesian Networks: The combination of knowledge and statistical data. In *Proceedings of the 10th conference on uncertainty in artificial intelligence*, (Ed. R. L. de Màntaras and D. Poole), 292-301.

Ho, T. K. (1995). Random decision forests. In *Proceedings of the 3rd International Conference on Document Analysis and Recognition*, 278-282.

Holder, L. B. (1995). Intermediate Decision Trees. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, 1056-1062.

Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI.

Holve, R. (1997). Rule generation for hierarchical fuzzy systems. In *Proceedings of the Annual Conference of the North American Fuzzy Information Processing Society*, 444-449.

Hosmer, D.W. and Lameshow, S. (1989). Applied Logistic Regression. New York: Wiley.

Janikow, C.Z. (1998). Fuzzy decision trees: Issues and methods. *IEEE Trans. Syst. Man and Cyb. Part B: Cybernetics*, **28**, 1-14.

Jensen, F.V. (1996). *An introduction to Bayesian Networks*. University College London Press, London.

Jordan, M.I. (1994). A Statistical Approach to Decision Tree Modelling, In M. Warmuth (Ed.), *Proceedings of the 7th Annual ACM Conference on Computational Learning Theory*. New York: ACM Press.

Kalles, D. and Morris, T. (1996). Efficient incremental induction of decision trees. *Machine Learning*, **24**, 231-242.

Kalousis, A. and Hilario, M. (2000). Supervised knowledge discovery from incomplete data. In *Proceedings of the 2nd International Conference on Data Mining 2000*. Cambridge, England. July 2000.WIT Press.

Kaplan, D. (1995). The impact of BIB spiralling-induced missing data patterns on goodness-of-fit tests in factor analysis. *Journal of Educational and Behavioral Statistics*, **20**, 69-82.

Kass, G.V. (1980). An exploratory technique for investigating large quantities of categorical data. *Applied Statistics*, **29**, 119-127.

Kendall, M. (1980). *Multivariate Analysis*. Charles Griffin & Company Ltd, London.

Keprta, S. (1996). Non-binary classification trees, *Statistics and Computing*, **6**, 231-243.

Kerber, R. (1992). Discretization of numeric attributes. In *Proceedings of the 15th International Conference on Machine Learning*, (144-151). San Francisco, CA: Morgan Kauffman.

Kim, J.-O. and Curry, J. (1977). The treatment of missing data in multivariate analysis. *Sociological Methods and Research*, **6**, 215-240.

Kirk, R.E. (1982). Experimental design (2nd Ed.). Monterey, CA: Brooks, Cole Publishing Company.

Kitchenham, B.A. (1996). A procedure for analysing unbalanced datasets, Keele University, Dept of Computer Science *Technical Report* TR10, ISSN 1353-7776.

Klir, G.J., and Folger, T.A. (1988). *Fuzzy Sets, Uncertainty, and Information.* Prentice-Hall, Englewood Cliffs, N.J.

Klockars, A.J., Hancock, G.R., and McAweeney, M.J. (1995). Power of unweighted and weighted versions of simultaneous and sequential multiple-comparison procedures. *Psychological Bulletin,* **118**, 300-307.

Kohavi, R. and Kunz, C. (1997). Option Decision Trees with majority votes. In D. Fisher, editor, *Machine Leaning Proceedings of the 14th International Conference.* Morgan Kaufmann.

Kononenko, Bratko, I., and Roscar, E. (1984). Experiments in automatic learning of medical diagnostic rules. (Technical report) Jozef Stefan Institute, Ljubljana, Yugoslavia.

Kononenko, I. (1991). Semi-naïve Bayesian classifier. In *Proceedings of European Conference on Artificial Intelligence*, 206-219.

Koza, J.R. (1992). *Advances in Genetic Programming.* MIT Press, Cambridge, MA.

Krzanowski, W.J. (1990). *Principles of Multivariate Analysis.* Oxford, UK: Clarendon Press.

Lakshminarayan, K., Harp, S.A., Samad, T. (1999). Imputation of Missing Data in Industrial Databases. *Applied Intelligence*, **11**, 259-275.

Langley, P. (1993). Induction of recursive Bayesian classifiers. In *Proc. European Conf. on Machine Learning,* (153-164). Springer Verlag.

Langley, P. (1996). *Element of Machine Learning.* Morgan Kauffman Publisher, Inc., San Francisco, California.

Langley, P., Iba, W. and Thompson, K. (1992). An analysis of Bayesian classifiers. In *Tenth National Conference on Artificial Intelligence,* (223-228). San Jose, California.

Langley, P. and Sage, S (1994). Induction of selective Bayesian classifiers. In *Proc. Conf. on Uncertainty in AI*, Morgan Kauffmann.

Latour, D., Latour, K., and Wolfinger, R.D. (1994). Get-Started with PROC MIXED, Software Sales and Marketing Department, SAS Institute Inc., Cary, NC.

Li, X. and Dubes, R.C. (1986). Tree classifier design with permutation statistic. *Pattern Recognition*, **16**, 69-80.

Little, R.J.A. and Rubin, D.B. (1987). *Statistical Analysis with missing data*. New York: Wiley.

Little, RJ.A. and Rubin, D. B. (1990). The analysis of social science data with missing values. In J. Fox and J.S. Long (Eds.), Modern Methods of Data Analysis (375-409). London: Sage.

Little, R.J.A. and Schenker, N (1995). The analysis of social science data with missing values. *Social Methods and Research*, **18**, 292-326.

Little, R.J.A. and Vartivarian, S. (2003). On weighting the rates in nonresponse weights. *Statistics and Medicine*, **22**, 1589-1599.

Liu, W.Z., White, A.P., Thompson, S.G., and Bramer, M.A. (1997). Techniques for dealing with missing values in classification. In *Advances in Intelligent Data Analysis*; Lecture Notes in Artificial Intelligence, edited by X. Liu, P. Cohen and M. Berthold, 527-536.

Lobo, O.O. and Numao, M. (1999) Ordered Estimation of Missing Values. In *Proceedings of the 3rd Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Lecture Notes in Computer Science, 1574, 274-278.

Lobo O. O. and Numao, M. (2000). On the Applicability of a Machine Learning Method for Estimating Missing Values. *IMLC 2000*, Palo Alto, California.

Loh, W.-Y. and Vanichsetakul, N. (1988). Tree-structured classification via Generalised Discriminant Analysis. *Journal of the American Statistical Association*, **83**, 715-728.

Loh, W.-Y. and Shih, Y.-S. (1997). Split selection methods for classification trees. *Statistica Sinica*, **7**, 815-840.

López de Màntaras (1991). A Distance-Based Attribute Selection Measure for Decision Tree induction *Machine Learning*, **6**, 81-92.

Luzowski, A. (1996). Crisp rule extraction from perceptron network classifiers. In Volume of Plenary: Panel and Special Session, *International Conference on Neural Networks*, Washington DC.

Mardia, K.V., Kent, J.T., and Bibby, J.M. (1979). *Multivariate Analysis*. Harcourt Brace & Company, London.

McCullagh, P., and Nelder, J.A. (1990). Generalised Linear Models, 2nd Edition, Chapman and Hall, London, England.

McFadden, D. (1976). A Comment on Discriminant Analysis 'Versus' Logit Analysis, *Annals of Economic and Social Measurement*, **5**, 511-523.

McKee, T.E. (1995). Predicting Bankruptcy via Induction. *Journal of Information Technology*, **10**, 26-36.

McLachlan, G.J. (1992). *Discriminant Analysis and Statistical Pattern Recognition.* New York: John Wiley.

McLachlan, G.J. and Basford, K.E. (1988). *Mixture Models: Inference and Applications to Clustering.* New York: Marcel Dekker.

McLachlan, G.J. and Peel, D. (2000). *Finite Mixture Models.* New York: John Wiley.

Menard, S. (1995). *Applied Logistic Regression.* Sage Publications, Inc, Thousand Oaks, CA.

Meng, X. L. and Rubin, D.B. (1991). IPF for contingency tables with missing data via the ECM algorithm. Proceedings of the Statistical Computing Section, American Statistical Association, 244–247.

Miller, R.G. 1997. *Beyond ANOVA.* Chapman & Hall

Mingers, J. (1986). Expert Systems – experiments with rule induction. *Journal of the Operational Research Society*, **37**, 1031-1037

Mingers, J. (1989a). An empirical comparison of Pruning Methods for Decision-Tree Induction. *Machine Learning,* **3**, 227-243.

Mingers, J. (1989b). An empirical comparison of Selection Measures for Decision-Tree Induction. *Machine Learning,* **3**, 319-342.

Michalski, R.S., Mozetic, I., Hong, J., and Lavrac, N. (1986). The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. In *Proceedings of the 5th National Conference on Artificial Intelligence,* (1041-1045). Philadelphia, PA: AAAI Press.

Michie, D., Spiegelhalter, D., and Taylor, C. (1994). *Machine Learning, Neural and Statistical Classification*. Ellis Horwood.

Mitchell, T.M. (1997). *Machine Learning*, McGraw Hill.

MINITAB. (2002). MINITAB *Statistical Software for Windows 9.0*. MINITAB, Inc., PA, USA.

Monago, M.M. and Kodratoff, Y. (1987). Noise and knowledge acquisition. In J. McDermott editor, *IJCAI-87*, (348-354). Kaufmann, CA.

Morgan, J.N. and Sonquist, J.A. (1963). Problems in the analysis of survey data and a proposal. *Journal of the American Statistical Association*, **58**, 415-434.

Morgan, J.N. and Messenger, R.C. (1973). THAID- a sequential analysis program for the analysis of nominal scale dependent variables. Survey Research Center, University of Michigan.

Müller, W. and Wysotzki, F. (1994). Automatic Construction of Decision Trees for Classification. *Annals of Operations Research*, 231-247.

Müller, W. and Wysotzki, F. (1996). The Decision Tree Algorithm CAL5 Based on a Statistical Approach to its Splitting Algorithm Automatic Construction of Decision Trees for Classification. In Makhaeizadeh, G. and Taylor, C.C. (Eds.). *Machine Learning and Statistics*. The Interface. John Wiley and Sons. New York.

Murphy, O.J. and McCraw, R.L. (1991). Designing storage efficient decision trees. *IEEE Transactions on Computing*, **40** (3): 315-319.

Murthy, S.K. and Salzberg, S. (1992). Lookahead and pathology in decision tree induction. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, (309-347). Montreal, Canada: Morgan Kauffman.

Murthy, S., Kasif, S., and Biegel, R. (1993). OC1: Randomised induction of oblique decisions trees. In *Proceedings of the 11th National Conference on Artificial Intelligence*, 322-327.

Murthy, S., Kasif, S., and Salzberg, S. (1994). A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research*, **2**, 1-32.

Muthén, B.O., Kaplan, D., and Hollis, M. (1987). On structural equation modelling with data that are not missing completely at random. *Psychometrika*, **52**, 431-462.

Myrtveit, I., Stensrud, E., and Olsson, U. (2001). Analyzing Data Sets with Missing Data: An Empirical Evaluation of Imputation Methods and Likelihood-Based Methods. *IEEE Transactions on Software Engineering*, **27** (11), 1999-1013.

Niblett, T. (1987). Constructing decision trees in noisy domains. *Proceedings of the Second European Working Session on Learning*, 67-78. Bled. Yugoslavia: Sigma.

Niblett, T. and Bratko, I. (1986). Learning decision rules in noisy domains. In *Proceedings of Expert Systems*, (25-34). Cambridge, England: Cambridge University Press.

Oates, T and Jensen, D. (1998). Large datasets lead to overly complex models: an explanation and solution. In *Proc Fourth International Conference on Knowledge Discovery and Data Mining, (*294-298). AAI Press. New York City, New York.

Oliver, J.J., and Hand, D.J. (1993). On pruning and averaging decision trees. In *Proceedings of the 12th International Machine Learning Conference,* (430-437). Tahoe City, CA: Morgan Kauffman.

Olkin, I. and Tate, R.F. (1961). Multivariate correlation models with mixed discrete and continuous variables. *Annals of Mathematical Statistics*, **32**, 448-465.

Pagallo, G. and Haussler, D. (1990). Boolean feature discovery in empirical learning. *Machine Learning*, **5**, 71-100.

Patterson, D.W. (1996). *Artificial Neural Network: Theory and Practices.* Prentice-Hall, Singapore.

Pawlak, Z. (1991). *Rough Sets. Theoretical Aspects of Reasoning About Data.* Kluwer Academic Publishers, Dordretch, Boston, London.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kauffman, San Mateo, CA.

Pearl, J. (1994). Causal diagrams for empirical research (with discussion). *Biometrika*, **82**, 669-710.

Pedrycz, W. (1996). Data mining and fuzzy modelling. In *Proceedings of 1996 Biennale Conference of the Northern American Fuzzy Information Processing Society (NAFIS)*, 263-267.

Pinder, J.P. (1996). Decision Analysis Using Multinomial Logit Models: Mortgage Portfolio Valuation. *Journal of Economics and Business*, **48**, 66-77.

Press, S. (1989). *Bayesian Statistics*. New York, Wiley.

Pyle, D. (1999). *Data Preparation for Data Mining*. Morgan Kauffman, San Francisco.

Quinlan, J.R. (1979). Induction over Large Databases: Report HPP-79-14, Stanford University.

Quinlan, J.R. (1983). Learning efficient classification procedures and their application to chess and games. In RS Michalski, JG Carbonelli, and TM Mitchell (Eds.), *Machine Learning: An Artificial Intelligence approach*. Los Altos: Morgan Kauffman.

Quinlan, J.R. (1985). Decision tress and multi-level attributes. *Machine Intelligence*. Vol. 11, (Eds.). J. Hayes and D. Michie. Chichester England: Ellis Horwood.

Quinlan, J.R. (1986). Induction of decision trees, *Machine Learning*, 1, 81-106.

Quinlan, J.R. (1987). Simplifying decision trees, *International Journal of Man - Machine Studies*, **27**, 221-234.

Quinlan, J.R. (1988). Decision trees and multi-valued attributes, *Machine Intelligence*, **11**, 305-318.

Quinlan, J.R. (1989). Unknown attribute values in induction. In *Proceedings of the 6th International Machine Learning Workshop*, (164-168). Ithaca, NY.

Quinlan, J.R. (1993). *C.4.5: Programs for machine learning.* Los Altos, California: Morgan Kauffman Publishers, INC.

Quinlan, J.R. (2002) http://www.rulequest.com/see5-comparison.html

Quinlan, JR. and Rivest, R.L. (1989). Inferring decision trees using minimum description length principle. *Information and Computation Machine Learning*, **80** (3), 227-248.

Ripley, B.D. (1992). *Pattern Recognition and Neural Networks.* Cambridge University Press, New York: John Wiley.

Ripley, B.D. (1994). Neural networks and related methods for classification. *Journal of the Royal Statistical Society,* Series B **56** (3), 409-437.

Ripley, B.D. (1996). *Pattern Recognition and Neural Networks.* Cambridge: Cambridge University Press.

Robins, D.B. and Wang, N. (2000). Inference for imputation estimators. *Biometrika*, **87**, 113-124.

Roth, P.L. (1994). Missing data: A conceptual overview for applied psychologists. *Personnel Psychology*, **47**, 537-560.

Rounds, E. (1980). A combined non-parametric approach to feature selection and binary decision tree pattern, *Pattern Recognition*, **12**, 313-317.

Rubin, D.B. (1976). Inference and missing data. *Biometrika*, **61**, 581-592.

Rubin, D.B. (1987). *Multiple Imputation for Nonresponse in Surveys.* New York: John Wiley and Sons.

Rubin, D.B. and Little, R.J.A. (1986). *Statistical Analysis with Missing Data.* Chichester: John Wiley and Sons.

Rubin, D.B. and Schenker, N. (1986). Multiple Imputation for Interval Estimation From Simple Random Samples With Ignorable Nonresponse. *Journal of the American Statistical Association*, **81** (394), 366-374.

Rumelhart, D.E., Hinton, G.E., and Williams, R.J. (1986). Learning international representation by error propagation. In D.E. Rumelhart and J.L. McClelland (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition. Volume 1: Foundations.* Cambridge, MA: MIT Press.

Safavian, S.R. and Landgrebe, D. (1991). A survey of decision tree classifiers. *IEEE Transactions on Systems, Man and Cybernetics,* **21**, 660-74.

Sande, I.G. (1983). *Hot-Deck Imputation Procedures. Incomplete Data in Sample Surveys.* 3. New York: Academic Press.

Schaffer, C (1993). Overfitting avoidance as Bias. *Machine Learning,* **10**, 153-178.

Schafer, J.L. (1997). *Analysis of Incomplete Multivariate Data.* Chapman and Hall, London.

Schafer, J.L. and Olsen, M.K. (1998). Multiple Imputation for multivariate missing data problems: a data analyst's perspective. *Multivariate Behavioral Research,* **33** (4), 545-571.

Schlimmer, J.C. and Fisher, D. (1986). A case study of incremental concept induction. In *Proceedings of the 5th National Conference on Artificial Intelligence,* (496-501). San Mateo, CA: Morgan Kauffman.

Sentas, P., Lefteris, A., and Stamelos, I. (2004). Multiple Logistic Regression as Imputation method Applied on Software Effort prediction. In *Proc. of the 10th International Symposium on Software Metrics,* Chicago, 14-16 September 2004.

Shannon, C.E. (1948a). A mathematical theory of communication. *Bell Syst. Tech. J,* **27**, 379-423.

Shannon, C.E. (1948b). A mathematical theory of communication, Part 2. *Bell Syst. Tech. J,* **27**, 623-656.

Shannon, W. (1998). Averaging Classification Tree Models. In *Proceedings of the 30th Symposium on the Interface.*

Shannon, W., and Banks, D. (1998). Combining Classification Trees Using MLE. *Statistics and Medicine,* In Press.

Shapiro, A. (1987). *Structured Induction in Expert Systems.* Addison Wesley, London.

Shavlik, J.W., Mooney, R.J., and Towell, G.G. (1991). Symbolic and Neural Network Learning Algorithms: An Experimental Comparison. *Machine Learning*, **6**, 111-143.

Shih, Y-S. (1999). Families of splitting criteria for classification trees. *Statistics and Computing*, **9**, 309-315.

Silverman, B. (1986). *Density Estimation for Statistics and Data Analysis.* Chapman and Hall, New York.

Singh, M. (1997). Learning Bayesian Networks from incomplete data. In *AAAI '97*, 27-31.

Smith, J.W., Everhart, J.E., Dickson, W.C., Knowler, W.C., and Johannes, R.S. (1988). Using ADAP learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the Symposium on Computer Applications and Medical Care* (261-265). *IEEE Computer Society Press.*

Song, Q. and Sheppered, M. (2004). A Short Note on Safest Default Missingness Mechanism Assumptions. In *Empirical Software Engineering.* (Accepted for publication in 2004).

Strike, K., El-Emam, K.E., Madhavji, N. (2001). Software Cost Estimation with Incomplete Data. *IEEE Transaction on Software Engineering*, **27** (10), 890-908.

Stutz, J. and Cheeseman, P. (1995) AutoClass - a Bayesian Approach to Classification. In *Maximum Entropy and Bayesian Methods, Cambridge 1994,* John Skilling & Sibusiso Sibisi, Eds. Kluwer Academic Publishers, Dordrecht.

Sokal, R. and Rohlf, F. (1981). Biometry. San Francisco: Freeman.

SAS (2000). *SAS Version 8.0 for Windows.* SAS Institute, Inc., North Carolina, USA.

SPSS. (1997). *SPSS Missing Values Analysis 7.5.* Chicago: SPSS, Inc.

SPSS. (2002). *SPSS for Windows 11.0.* SPSS, Inc., Chicago, USA.

S-PLUS. (2003). *S-PLUS 6.2 for Windows*. MathSoft, Inc., Seattle, Washington, USA.

Tabachnick, B.G. and Fidell, L.S. (2001). *Using multivariate statistics* (4th Ed.). Needham Heights, MA: Allyn and Bacon.

Talmon, J.L. (1986). A multi-class non-parametric partitioning algorithm. *Pattern Recognition Letters*, **4**, 31-38.

Tanner, M.A. and Wong, W.H. (1987). The calculation of Posterior Distributions by Data Augmentation (with discussion). *Journal of the American Statistical Association*, **82**, 528-550

Taylor, P.C. and Silverman, B.W. (1993). Block diagrams and splitting criteria for classification trees. *Statistics and Computing*, **3** (4), 147-161.

Therneau, T.M., and Atkinson, E.J. (1997). An Introduction to Recursive Partitioning Using the RPART Routines. *Technical Report*, Mayo Foundation.

Tresp, V., Neuneier, R., and Ahmad, S. (1995). Efficient Methods for Dealing with Missing Data in Supervised Learning. *Advances in Neural Information processing Systems 7* (Eds. Tesauro, G., Touretzky, D.S., and Leen, T.K.), MIT Press, Cambridge, MA.

Utgoff, P. (1986). *Machine Learning of Inductive Bias*. Kluwer Academic Publishers.

Utgoff, P. (1991). Incremental induction of decision trees. *Machine Learning*, **4**, 161-186.

Vadera, S. and Nechab, S. (1994). Id3, its children and their safety. *BCS Specialist Group on Expert Systems Newsletter*, **31**, 11-21.

Vach, W. (1995). Classification Trees. *Computational Statistics*, **10**, 9-14.

Van de Merckt, T. (1993). Decision trees in numerical attribute spaces. In *Proceedings of the 13th IJCAI*, (1016-1021). Chambery, France.

Vapkin, V.N. (1995). *The Nature of Statistical Learning Theory*. Springer – Verlag, New York.

Venables, W.N. and Ripley, B.D. (1994). *Modern Applied Statistics with S-PLUS*. New York: Springer.

Wallace, C.S. and Patrick, J.D. (1993) Coding decision tress. *Machine Learning*, **7**, 279-292.

Wallace, C.S. (1998) *Intrinsic classification of spatially correlated data*, in C J van Rijsbergen (ed), Computer Journal, Vol. **41**, No 8, Oxford University Press, UK, ISSN: 0010-4620, 602-611

Wand, M.P. and Jones, M.C. (1995). *Kernel Smoothing*. London: Chapman and Hall.

Weiss, S., Galen, R. and Tadepelli, P. (1990). Maximizing the predictive rule of production rules. *Artificial Intelligence*. **45**, 47-71.

White, AP. (1987). Probabilistic Induction by dynamic path generation in virtual trees: In Brammer, Max (ed.) *Research & Development in Expert Systems:* Cambridge: Cambridge University Press.

White, A.P. (2001). Private communication.

White, A.P. and Liu, W.Z. (1994). Bias in information-based measures in decision tree induction. *Machine Leaning*, **15**, 321-329.

Wilks, S.S. (1932). Moments and distributions of estimates of population parameters from fragmentary samples. *Annals of Mathematical Statistics*, **3**, 163-195.

Winston, P. (1992). *Artificial Intelligence.* Addison-Wesley, third edition. Part II: Learning and regularity Recognition.

Wishart, D. (1999). Clustering methods for Large Data Problems. In *Bulletin of the International Statistical Institute, Proceedings, Book 1*, 437-440.

Wong, A.K.C. and Chiu, D.K.Y. (1987). An event- covering method for effective probabilistic inference. *Pattern Recognition*, **20**, 2, 245-255.

Wu, C.F.J. (1983). On the convergence of the EM algorithm. *The Annals of Statistics*, **11**, 95-103.

Yun, S.Q. and Fu, K.S. (1983). A method for the design of binary tree classifiers. *Proc. 3rd Symp. Machine Processing of Remotely Sensed* Data, Purdue Univ.

Zadeh, L.A. (1965). Fuzzy sets. *Information control,* **8**, 338-353.

Zadeh, L.A. (1994). Soft computing and fuzzy logic. *IEEE Software,* 48-56.

Zhang, H. (1998). Classification Trees for Multiple Binary Responses. *Journal of the American Statistical Association*, **93**, 441, 180-193.

Zheng, Z. and Webb, G.I. (1997). Lazy Bayesian Trees. *Technical Report* (TR C97/07). Deakin University, Australia.

Zheng, Z. and Low, B.T. (1999). Classifying Unseen Cases with Many Missing Values. *Technical Report* (TR C90/02). Deakin University, Australia.

Appendix

APPENDIX - ANALYSIS OF VARIANCE TABLES

Table 4.3 Analysis of Variance for significance tests for existing training and testing methods

Source of variation	*Sum of Squares*	*Degrees of freedom*	*Mean-Square*	*F-ratio*	*p -value*
Main effects:					
A	0.450932	6	0.075155	76.07	0.000
B	0.147939	1	0.147939	10.85	0.004
C	1.058058	2	0.529029	671.26	0.000
D	4.403085	2	2.201542	619.45	0.000
E	1.128079	20	0.056404	3.42	0.001
Two-way interactions:					
AB	0.039702	6	0.006617	8.81	0.000
AC	0.003633	12	0.000303	3.32	0.000
AD	0.016058	12	0.001338	3.80	0.000
AE	0.118562	120	0.000988	1.23	0.131
BC	0.003828	2	0.001914	7.60	0.002
BD	0.004593	2	0.002296	1.78	0.181
BE	0.272686	20	0.013634	7.75	0.000
CD	0.000415	4	0.000104	0.31	0.869
CE	0.031524	40	0.000788	2.11	0.011
DE	0.142162	40	0.003554	2.47	0.002
Three-way interactions:					
ABC	0.000214	12	0.000018	0.23	0.997
ABD	0.003027	12	0.000252	0.82	0.629
ABE	0.090102	120	0.000751	2.37	0.000
ACD	0.001968	24	0.000082	1.08	0.363

Table 4.3 Analysis of Variance for significance tests for existing training and testing methods continued

Three-way interactions:					
ACE	0.021911	240	0.000091	1.07	0.317
ADE	0.084522	240	0.000352	1.12	0.195
BCD	0.000561	4	0.000140	0.65	0.628
BCE	0.010076	40	0.000252	1.12	0.326
BDE	0.051507	40	0.001288	2.83	0.000
CDE	0.026534	80	0.000332	1.49	0.038
Four-way interactions:					
ABCD	0.001281	24	0.000053	0.78	0.762
ABCE	0.018586	240	0.000077	1.13	0.129
ABDE	0.073783	240	0.000307	4.50	0.000
ACDE	0.036461	480	0.000076	1.11	0.125
BCDE	0.017244	80	0.000216	0.06	0.000
Five-way interactions:					
ABCDE	0.0.32818	480	0.000068		
Total	8.291848	2645			

Note: A = training and testing methods (repeated measures factor with 7 levels)

B = number of attributes with missing values (repeated measures factor with 2 levels)

C = missing data proportions (repeated measures factor with 3 levels)

D = missing data mechanisms (repeated measures factor with 3 levels)

E = datasets (random effects with 21 levels)

p-values below 0.01 indicate statistically significant effects at the 1% significance level.

Table 4.4 Analysis of Variance for significance tests for existing training methods

Source of variation	*Sum of Squares*	*Degrees of freedom*	*Mean-Square*	*F-ratio*	*p -value*
Main effects:					
A	0.479026	7	0.068432	82.63	0.000
B	0.261300	1	0.261300	96.34	0.000
C	1.014373	2	0.507187	611.28	0.000
D	3.948964	2	1.974482	540.47	0.000
E	0.474797	20	0.023740	4.41	
Two-way interactions:					
AB	0.040945	7	0.005849	5.28	0.000
AC	0.006552	14	0.000468	6.02	0.000
AD	0.013805	14	0.000986	4.65	0.000
AE	0.115948	140	0.000828	0.72	0.974
BC	0.008107	2	0.004054	8.86	0.001
BD	0.018664	2	0.009332	13.55	0.000
BE	0.054247	20	0.002712	1.42	0.122
CD	0.001321	4	0.000330	0.66	0.621
CE	0.033189	40	0.000830	1.03	0.453
DE	0.146130	40	0.003653	3.43	0.000
Three-way interactions:					
ABC	0.001184	14	0.000085	1.28	0.220
ABD	0.004249	14	0.000303	1.71	0.053
ABE	0.155147	140	0.001108	5.80	0.000
ACD	0.001939	28	0.000069	1.21	0.212
ACE	0.021751	280	0.000078	1.09	0.256
ADE	0.059381	280	0.000212	1.16	0.105
BCD	0.001341	4	0.000335	2.17	0.080
BCE	0.018307	40	0.000458	2.72	0.000

Table 4.4 Analysis of Variance for significance tests for existing training methods continued

Three-way interactions:					
BDE	0.027556	40	0.000689	2.46	0.000
CDE	0.039955	80	0.000499	3.13	0.000
Four-way interactions:					
ABCD	0.002378	28	0.000085	1.63	0.023
ABCE	0.018516	280	0.000066	1.27	0.010
ABDE	0.049621	280	0.000177	3.40	0.000
ACDE	0.032029	560	0.000057	1.10	0.140
BCDE	0.012370	80	0.000155	2.96	0.000
Five-way interaction:					
ABCDE	0.029228	560	0.000052		
Total	7.092320	3023			

Note: A = training methods (repeated measures factor with 8 levels)

B = number of attributes with missing values (repeated measures factor with 2 levels)

C = missing data proportions (repeated measures factor with 3 levels)

D = missing data mechanisms (repeated measures factor with 3 levels)

E = datasets (random effects with 21 levels)

p-values below 0.01 indicate statistically significant effects at the 1% significance level.

Table 4.5 Analysis of Variance for significance tests for existing testing methods

Source of variation	*Sum of Squares*	*Degrees of freedom*	*Mean-Square*	*F-ratio*	*p -value*
Main effects:					
A	0.536685	6	0.089447	81.28	0.000
B	0.139581	1	0.139581	73.92	0.000
C	1.028421	2	0.514211	722.59	0.000
D	3.584060	2	1.792030	546.29	0.000
E	0.784423	20	0.0.9221	7.89	0.000
Two-way interactions:					
AB	0.020471	6	0.003412	5.79	0.000
AC	0.004935	12	0.000411	4.63	0.000
AD	0.013375	12	0.001115	5.81	0.000
AE	0.132060	120	0.001100	1.73	0.001
BC	0.007683	2	0.003842	8.78	0.001
BD	0.002380	2	0.001190	1.81	0.177
BE	0.037765	20	0.001888	1.39	0.147
CD	0.001417	4	0.000354	0.80	0.531
CE	0.028465	40	0.000712	1.02	0.460
DE	0.131214	40	0.003280	3.34	0.000
Three-way interactions:					
ABC	0.000572	12	0.000048	0.55	0.883
ABD	0.002085	12	0.000174	1.37	0.180
ABE	0.070712	120	0.000589	3.71	0.000
ACD	0.002492	24	0.000104	1.36	0.122
ACE	0.021316	240	0.000089	0.82	0.940
ADE	0.046081	240	0.000192	1.30	0.020
BCD	0.000292	4	0.000073	0.44	0.781
BCE	0.017493	40	0.000437	2.20	0.001

Table 4.5 Analysis of Variance for significance tests for existing testing methods continued

Three-way interactions:					
BDE	0.026347	40	0.000659	2.76	0.000
CDE	0.035612	80	0.000445	2.37	0.000
Four-way interactions:					
ABCD	0.002218	24	0.000092	1.68	0.024
ABCE	0.002218	240	0.000092	1.58	0.000
ABDE	0.002218	240	0.000092	2.30	0.000
ACDE	0.002218	480	0.000092	1.39	0.000
BCDE	0.002218	80	0.000092	3.02	0.000
Fiver-way interaction:					
ABCDE	0.002218	480	0.000092		
Total	60806048	2645			

Note: A = testing methods (repeated measures factor with 7 levels)

B = number of attributes with missing values (repeated measures factor with 2 levels)

C = missing data proportions (repeated measures factor with 3 levels)

D = missing data mechanisms (repeated measures factor with 3 levels)

E = datasets (random effects with 21 levels)

p-values below 0.01 indicate statistically significant effects at the 1% significance level.

Table 4.6 (a) Analysis of Variance for significance tests for existing testing methods (for unbalanced data)

Source of variation	*Adjusted Sum of Squares*	*Degrees of freedom*	*Adjusted Mean-Square*	*F-ratio*	*p -value*
Main effects:					
A	0.424224	6	0.070704	419.62	0.000
B	0.119643	1	0.119643	710.07	0.000
C	0.854400	2	0.427200	2535.40	0.000
D	2.697587	2	1.384794	8005.00	0.000
N	0.037574	1	0.037574	223.00	0.000
E(N)	0.746849	19	0.039308	233.29	0.000
Two-way interactions:					
AN	0.009559	6	0.001593	9.46	0.000
BN	0.000379	1	0.000379	2.25	0.000
CN	0.000363	2	0.000182	1.08	0.340
DN	0.028265	2	0.014132	83.87	0.000
AE(N)	0.122501	114	0.001075	6.38	0.000
BE(N)	0.037386	19	0.001968	11.68	0.000
CE(N)	0.028101	38	0.000740	4.39	0.000
DE(N)	0.102949	38	0.002709	16.08	0.000
Residual	0.403374	2394	0.000168		
Total	5.613154	2645			

Note: A = testing methods (repeated measures factor with 7 levels)

B = number of attributes with missing values (repeated measures factor with 2 levels)

C = missing data proportions (repeated measures factor with 3 levels)

D = missing data mechanisms (repeated measures factor with 3 levels)

N = binary attribute (repeated factor with 2 levels)

E(N) = dataset nested within binary attribute variable

p-values below 0.01 indicate statistically significant effects at the 1% significance level.

Table 4.6 (b) Analysis of Variance for significance tests for existing testing methods (for unbalanced data)

Source of variation	*Adjusted Sum of Squares*	*Degrees of freedom*	*Adjusted Mean-Square*	*F-ratio*	*p -value*
Main effects:					
A	0.424224	6	0.070704	65.80	0.000
B	0.119643	1	0.119643	60.80	0.000
C	0.854400	2	0.427200	577.68	0.000
D	2.697587	2	1.384794	497.86	0.000
Two-way interactions:					
AN	0.009559	6	0.001593	1.48	0.190
BN	0.000379	1	0.000379	0.19	0.666
CN	0.000363	2	0.000182	0.25	0.783
DN	0.028265	2	0.014132	5.22	0.010
Residual					
*AE(N)**	0.122501	114	0.001075		
BE(N+	0.037386	19	0.001968		
*CE(N)*x	0.028101	38	0.000740		
DE(N)#	0.102949	38	0.002709		

Note: * = error term for A and AN

\+ = error term for B and BN

x = error term for C and CN

= error term for D and DN

p-values below 0.01 indicate statistically significant effects at the 1% significance level.

Table 5.2 Analysis of Variance for significance tests for existing and new training and testing methods

Source of variation	*Sum of Squares*	*Degrees of freedom*	*Mean-Square*	*F-ratio*	*p -value*
Main effects:					
A	0.028279	2	0.014139	8.00	0.001
B	0.043457	1	0.043457	14.70	0.001
C	0.391570	2	0.795785	686.72	0.000
D	1.523519	2	0.761760	654.93	0.000
E	0.359481	20	0.017974	3.56	0.000
Two-way interactions:					
AB	0.007878	2	0.003939	6.16	0.005
AC	0.002653	4	0.000663	7.22	0.000
AD	0.010129	4	0.002532	10.72	0.000
AE	0.070681	40	0.007167	2.35	0.002
BC	0.002322	2	0.001161	19.36	0.000
BD	0.006586	2	0.003293	12.61	0.000
BE	0.059138	20	0.002957	4.13	0.000
CD	0.000648	4	0.000162	1.14	0.343
CE	0.011404	40	0.000285	2.10	0.020
DE	0.046525	40	0.001163	2.90	0.000
Three-way interactions:					
ABC	0.000075	4	0.000019	0.29	0.882
ABD	0.000570	4	0.000142	0.99	0.421
ABE	0.025568	40	0.000639	4.06	0.000
ACD	0.000769	8	0.000096	1.65	0.113
ACE	0.007347	80	0.000092	1.29	0.153
ADE	0.018898	80	0.000236	1.56	0.026
BCD	0.000144	4	0.000036	0.41	0.798
BCE	0.002399	40	0.00060	0.60	0.956

Table 5.2 Analysis of Variance for significance tests for existing and new training and testing methods continued

Three-way interactions:					
BDE	0.010450	40	0.000261	1.45	0.077
CDE	0.011355	80	0.000142	1.51	0.042
Four-way interactions:					
ABCD	0.000259	8	0.000032	0.64	0.747
ABCE	0.005115	80	0.000064	1.25	0.116
ABDE	0.011571	80	0.000145	2.83	0.000
ACDE	0.009291	160	0.000058	1.14	0.208
BCDE	0.006965	80	0.000087	1.71	0.002
Five-way interactions:					
ABCDE	0.008168	160	0.000051		
Total	2.683216	1133			

Note: A = existing and new training and testing methods (repeated measures factor with 3 levels)

B = number of attributes with missing values (repeated measures factor with 2 levels)

C = missing data proportions (repeated measures factor with 3 levels)

D = missing data mechanisms (repeated measures factor with 3 levels)

E = datasets (random effects with 21 levels)

p-values below 0.01 indicate statistically significant effects at the 1% significance level.

Table 6.2 Analysis of Variance for significance tests for existing and new testing methods

Source of variation	*Sum of Squares*	*Degrees of freedom*	*Mean-Square*	*F-ratio*	*p -value*
Main effects:					
A	0.252642	4	0.063160	43.58	0.000
B	0.065968	1	0.065968	36.41	0.000
C	0.695467	2	0.347733	701.64	0.000
D	2.270073	2	1.135037	633.93	0.000
E	0.584567	20	0.029228	7.18	0.000
Two-way interactions:					
AB	0.004742	4	0.001186	2.04	0.096
AC	0.002076	8	0.000260	3.75	0.000
AD	0.008282	8	0.001035	3.22	0.002
AE	0.115938	80	0.001449	2.38	0.000
BC	0.002447	2	0.001224	4.99	0.012
BD	0.000497	2	0.000248	0.50	0.612
BE	0.036232	20	0.001812	2.01	0.019
CD	0.007870	4	0.001968	8.11	0.000
CE	0.019824	40	0.000496	1.39	0.134
DE	0.071619	40	0.001790	2.75	0.000
Three-way interactions:					
ABC	0.000146	8	0.000018	0.26	0.977
ABD	0.002570	8	0.000321	1.14	0.337
ABE	0.046415	80	0.000580	1.91	0.000
ACD	0.002997	16	0.000187	3.27	0.000
ACE	0.011082	160	0.000069	0.87	0.811
ADE	0.051386	160	0.000321	1.10	0.270
BCD	0.000125	4	0.000031	0.26	0.902
BCE	0.009809	40	0.000245	1.72	0.017

Table 6.2 Analysis of Variance for significance tests for existing and new testing methods continued

Three-way interactions:					
BDE	0.019998	40	0.000500	1.41	0.067
CDE	0.019410	80	0.000243	1.86	0.002
Four-way interactions:					
ABCD	0.000888	16	0.000055	1.19	0.273
ABCE	0.011083	160	0.000069	1.49	0.002
ABDE	0.044954	160	0.000281	6.03	0.000
ACDE	0.018358	320	0.000057	1.23	0.032
BCDE	0.009583	80	0.000120	2.57	0.000
Five-way interactions:					
ABCDE	0.014909	320	0.000047		
Total					

Note: A = existing and new testing methods (repeated measures factor with 5 levels)

B = number of attributes with missing values (repeated measures factor with 2 levels)

C = missing data proportions (repeated measures factor with 3 levels)

D = missing data mechanisms (repeated measures factor with 3 levels)

E = datasets (random effects with 21 levels)

p-values below 0.01 indicate statistically significant effects at the 1% significance level.

Table 7.2 Analysis of Variance for significance tests for ensemble and missing data methods

Source of variation	*Sum of Squares*	*Degrees of freedom*	*Mean-Square*	*F-ratio*	*p-value*
Main effects:					
A	0.044556	3	0.014852	7.13	0.000
B	0.111257	1	0.111257	74.03	0.000
C	0.527178	2	0.263589	1061.86	0.000
D	1.915321	2	0.957661	750.31	0.000
E	0.369428	20	0.018471	5.11	0.000
Two-way interactions:					
AB	0.011480	3	0.003827	4.08	0.010
AC	0.002627	6	0.000438	3.27	0.005
AD	0.006761	6	0.001127	4.77	0.000
AE	0.124956	60	0.002083	2.00	0.003
BC	0.002988	2	0.001494	9.44	0.000
BD	0.004389	2	0.002195	10.79	0.000
BE	0.030057	20	0.001503	1.45	0.134
CD	0.002779	4	0.000695	4.21	0.004
CE	0.009929	40	0.000248	0.84	0.724
DE	0.051054	40	0.001276	3.52	0.000
Three-way interactions:					
ABC	0.000516	6	0.000086	0.91	0.487
ABD	0.001787	6	0.000298	1.70	0.126
ABE	0.056220	60	0.000937	4.93	0.000
ACD	0.001098	12	0.000092	1.19	0.293
ACE	0.016071	120	0.000134	1.45	0.046
ADE	0.028362	120	0.000236	1.36	0.056
BCD	0.000451	4	0.000113	1.64	0.173
BCE	0.006331	40	0.000158	1.89	0.020

Table 7.2 Analysis of Variance for significance tests for ensemble and missing data methods continued

Three-way interactions:					
BDE	0.008139	40	0.000203	1.23	0.211
CDE	0.013213	80	0.000165	2.46	0.001
Four-way interactions:					
ABCD	0.000592	12	0.000049	0.63	0.820
ABCE	0.011279	120	0.000094	1.19	0.129
ABDE	0.020999	120	0.000175	2.22	0.000
ACDE	0.018515	240	0.000077	0.98	0.570
BCDE	0.005503	80	0.000069	0.87	0.762
Five-way interactions:					
ABCDE	0.018942	240	0.000079		
Total					

Note: A = existing and ensemble missing data methods (repeated measures factor with 4 levels)

B = number of attributes with missing values (repeated measures factor with 2 levels)

C = missing data proportions (repeated measures factor with 3 levels)

D = missing data mechanisms (repeated measures factor with 3 levels)

E = datasets (random effects with 21 levels)

p-values below 0.01 indicate statistically significant effects at the 1% significance level.